Package 'ggalign'

November 14, 2024

Title A 'ggplot2' Extension for Consistent Axis Alignment

Version 0.0.5

Description A 'ggplot2' extension offers various tools for organizing and arranging plots. It is designed to consistently align a specific axis across multiple 'ggplot' objects, making it especially useful for plots requiring data order manipulation. A typical use case includes organizing combinations like a dendrogram and a heatmap.

License MIT + file LICENSE

URL https://github.com/Yunuuuu/ggalign, https://yunuuuu.github.io/ggalign/

BugReports https://github.com/Yunuuuu/ggalign/issues

Depends ggplot2 (>= 3.3.0)

Imports cli, grid, gtable, lifecycle, methods, rlang (>= 1.1.0), stats, utils, vctrs (>= 0.5.0)

Suggests bookdown, ggrastr, gridGraphics, knitr, patchwork, ragg, rmarkdown, scales, testthat (>= 3.0.0), vdiffr

ByteCompile true

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

VignetteBuilder knitr

NeedsCompilation no

Author Yun Peng [aut, cre] (<https://orcid.org/0000-0003-2801-3332>)

Maintainer Yun Peng <yunyunp96@163.com>

Repository CRAN

Date/Publication 2024-11-14 08:00:02 UTC

Contents

`active` *Plot Adding Context Settings*

Description

[Experimental]

These settings control the behavior of the plot when added to a layout, as well as the arrangement of individual plot areas within the layout.

Usage

```
active(order = waiver(), use = waiver(), name = waiver())
```

Arguments

`order`	An integer specifying the order of the plot area within the layout.
`use`	A logical (`TRUE`/`FALSE`) indicating whether to set the active context to the current plot when added to a layout. If `TRUE`, any subsequent `ggplot` elements will be applied to this plot.
`name`	A string specifying the plot's name, useful for switching active contexts through the `what` argument in functions like `quad_anno()`/`stack_switch()`.

Details

By default, the active context is set only for functions that add plot areas. This allows other ggplot2 elements-such as geoms, stats, scales, or themes- to be seamlessly added to the current plot area.

The default ordering of the plot areas is from top to bottom or from left to right, depending on the layout orientation. However, users can customize this order using the order argument.

---

align_dendro *Reorder or Group observations based on hierarchical clustering*

Description

[Stable]

This function aligns observations within the layout according to a hierarchical clustering tree, enabling reordering or grouping of elements based on clustering results.

Usage

```
align_dendro(
  mapping = aes(),
  ...,
  distance = "euclidean",
  method = "complete",
  use_missing = "pairwise.complete.obs",
  reorder_dendrogram = FALSE,
  merge_dendrogram = FALSE,
  reorder_group = FALSE,
  k = NULL,
  h = NULL,
  cutree = NULL,
  plot_dendrogram = TRUE,
  plot_cut_height = NULL,
  root = NULL,
  center = FALSE,
  type = "rectangle",
  size = NULL,
  data = NULL,
  no_axes = NULL,
  active = NULL,
  free_guides = deprecated(),
  free_spaces = deprecated(),
  plot_data = deprecated(),
  theme = deprecated(),
  free_labs = deprecated(),
  set_context = deprecated(),
  order = deprecated(),
  name = deprecated()
)
```

Arguments

mapping	Default list of aesthetic mappings to use for plot. If not specified, must be supplied in each layer added to the plot.
...	<dyn-dots> Additional arguments passed to geom_segment().
distance	A string of distance measure to be used. This must be one of "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski". Correlation coefficient can be also used, including "pearson", "spearman" or "kendall". In this way, 1 - cor will be used as the distance. In addition, you can also provide a dist object directly or a function return a dist object. Use NULL, if you don't want to calculate the distance.
method	A string of the agglomeration method to be used. This should be (an unambiguous abbreviation of) one of "ward.D", "ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (= WPGMA), "median" (= WPGMC) or "centroid" (= UPGMC). You can also provide a function which accepts the calculated distance (or the input matrix if distance is NULL) and returns a hclust object. Alternative, you can supply an object which can be coerced to hclust.
use_missing	An optional character string giving a method for computing covariances in the presence of missing values. This must be (an abbreviation of) one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs". Only used when distance is a correlation coefficient string.
reorder_dendrogram	A single boolean value indicating whether to reorder the dendrogram based on the means. Alternatively, you can provide a custom function that accepts an hclust object and the data used to generate the tree, returning either an hclust or dendrogram object. Default is FALSE.
merge_dendrogram	A single boolean value, indicates whether we should merge multiple dendrograms, only used when previous groups have been established. Default: FALSE.
reorder_group	A single boolean value, indicates whether we should do Hierarchical Clustering between groups, only used when previous groups have been established. Default: FALSE.
k	An integer scalar indicates the desired number of groups.
h	A numeric scalar indicates heights where the tree should be cut.
cutree	A function used to cut the hclust tree. It should accept four arguments: the hclust tree object, distance (only applicable when method is a string or a function for performing hierarchical clustering), k (the number of clusters), and h (the height at which to cut the tree). By default, cutree() is used.
plot_dendrogram	A boolean value indicates whether plot the dendrogram tree.
plot_cut_height	A boolean value indicates whether plot the cut height.
root	A length one string or numeric indicates the root branch.

center	A boolean value. if TRUE, nodes are plotted centered with respect to the leaves in the branch. Otherwise (default), plot them in the middle of all direct child nodes.
type	A string indicates the plot type, "rectangle" or "triangle".
size	The relative size of the plot, can be specified as a unit.
data	A matrix-like object. By default, it inherits from the layout matrix.
no_axes	**[Experimental]** Logical; if TRUE, removes axes elements for the alignment axis using theme_no_axes(). By default, will controled by the option- "ggalign.align_no_axes".
active	A active() object that defines the context settings when added to a layout.
free_guides	**[Superseded]** Please use plot_align() function instead.
free_spaces	**[Deprecated]** Please use plot_align() function instead.
plot_data	**[Deprecated]** Please use plot_data() function instead.
theme	**[Deprecated]** Please use plot_theme() function instead.
free_labs	**[Deprecated]** Please use plot_align() function instead.
set_context	**[Deprecated]** Please use active argument instead.
order	**[Deprecated]** Please use active argument instead.
name	**[Deprecated]** Please use active argument instead.

Value

A "AlignDendro" object.

ggplot2 specification

align_dendro initializes a ggplot data and mapping.

The internal will always use a default mapping of aes(x = .data$x, y = .data$y).

The default ggplot data is the node coordinates with edge data attached in ggalign attribute, in addition, a geom_segment layer with a data of the edge coordinates will be added.

node and tree segments edge coordinates contains following columns:

- index: the original index in the tree for the current node
- label: node label text
- x and y: x-axis and y-axis coordinates for current node or the start node of the current edge.
- xend and yend: the x-axis and y-axis coordinates of the terminal node for current edge.
- branch: which branch current node or edge is. You can use this column to color different groups.
- panel: which panel current node is, if we split the plot into panel using facet_grid, this column will show which panel current node or edge is from. Note: some nodes may fall outside panel (between two panel), so there are possible NA values in this column.
- .panel: Similar with panel column, but always give the correct branch for usage of the ggplot facet.
- panel1 and panel2: The panel1 and panel2 variables have the same functionality as panel, but they are specifically for the edge data and correspond to both nodes of each edge.
- leaf: A logical value indicates whether current node is a leaf.

Axis Alignment for Observations

It is important to note that we consider rows as observations, meaning `vec_size(data)`/`NROW(data)` must match the number of observations along the axis used for alignment (x-axis for a vertical stack layout, y-axis for a horizontal stack layout).

- `quad_layout()`/`ggheatmap()`: For column annotation, the layout `matrix` will be transposed before use (if `data` is a function, it is applied to the transposed matrix), as column annotation uses columns as observations but alignment requires rows.
- `stack_layout()`: The layout matrix is used as is, aligning all plots along a single axis.

See Also

- `dendrogram_data()`
- `hclust2()`

Examples

```
ggheatmap(matrix(rnorm(81), nrow = 9)) +
    anno_top() +
    align_dendro()
ggheatmap(matrix(rnorm(81), nrow = 9)) +
    anno_top() +
    align_dendro(k = 3L)
```

---

align_gg *Create ggplot object*

Description

[Stable]

`align_gg()` is similar to `ggplot` in that it initializes a `ggplot` data and `mapping`. Same with other `align_*` functions. `align_gg()` allowing you to provide data in various formats, including matrices, data frames, or simple vectors. By default, it will inherit from the layout. If a function, it will apply with the layout matrix. `ggalign` is an alias of `align_gg`.

Usage

```
align_gg(
  mapping = aes(),
  size = NULL,
  data = waiver(),
  limits = TRUE,
  facet = TRUE,
  no_axes = NULL,
  active = NULL,
  set_context = deprecated(),
  order = deprecated(),
```

```
  name = deprecated(),
  free_guides = deprecated(),
  free_spaces = deprecated(),
  plot_data = deprecated(),
  theme = deprecated(),
  free_labs = deprecated()
)
```

Arguments

mapping	Default list of aesthetic mappings to use for plot. If not specified, must be supplied in each layer added to the plot.
size	The relative size of the plot, can be specified as a unit.
data	A flexible input that specifies the data to be used • NULL: No data is set. • waiver(): Uses the layout matrix. • A function (including purrr-like lambda syntax) that is applied to the layout matrix, and must return a matrix. If you want to transform the final plot data, please use plot_data(). • A matrix, data frame, or atomic vector.
limits	Logical; if TRUE, sets layout limits for the plot.
facet	Logical; if TRUE, applies facets to the layout. If FALSE, limits will also be set to FALSE.
no_axes	**[Experimental]** Logical; if TRUE, removes axes elements for the alignment axis using theme_no_axes(). By default, will controled by the option- "ggalign.align_no_axes".
active	A active() object that defines the context settings when added to a layout.
set_context	**[Deprecated]** Please use active argument instead.
order	**[Deprecated]** Please use active argument instead.
name	**[Deprecated]** Please use active argument instead.
free_guides	**[Superseded]** Please use plot_align() function instead.
free_spaces	**[Deprecated]** Please use plot_align() function instead.
plot_data	**[Deprecated]** Please use plot_data() function instead.
theme	**[Deprecated]** Please use plot_theme() function instead.
free_labs	**[Deprecated]** Please use plot_align() function instead.

Value

A "AlignGG" object.

ggplot2 specification

`align_gg` initializes a ggplot `data` and `mapping`.

`align_gg()` always applies a default mapping for the axis of the data index in the layout. This mapping is `aes(y = .data$.y)` for horizontal stack layout (including left and right annotation) and `aes(x = .data$.x)` for vertical stack layout (including top and bottom annotation).

The data in the underlying `ggplot` object will contain following columns:

- `.panel`: the panel for the aligned axis. It means `x-axis` for vertical stack layout (including top and bottom annotation), `y-axis` for horizontal stack layout (including left and right annotation).
- `.x` or `.y`: the `x` or `y` coordinates
- `.names` (`vec_names()`) and `.index` (`vec_size()`/`NROW()`): A factor of the names (only applicable when names exists) and an integer of index of the original data.
- `.row_names` and `.row_index`: the row names and an integer of row index of the original matrix (only applicable if `data` is a `matrix`).
- `.column_names` and `.column_index`: the column names and column index of the original matrix (only applicable if `data` is a `matrix`).
- `value`: the actual value (only applicable if `data` is a `matrix` or atomic vector).

In the case where the input data is already a data frame, 4 additional columns (`.x`/`.y`, `.names`, `.index`, and `.panel`) are added to the data frame.

It is recommended to use `.x`/`.y`, or `.names` as the `x`/`y` mapping.

If the data inherits from `quad_layout()`/`ggheatmap()`, an additional column will be added.

- `.extra_panel`: the panel information for column (left or right annotation) or row (top or bottom annotation).

Axis Alignment for Observations

It is important to note that we consider rows as observations, meaning `vec_size(data)`/`NROW(data)` must match the number of observations along the axis used for alignment (x-axis for a vertical stack layout, y-axis for a horizontal stack layout).

- `quad_layout()`/`ggheatmap()`: For column annotation, the layout `matrix` will be transposed before use (if `data` is a function, it is applied to the transposed matrix), as column annotation uses columns as observations but alignment requires rows.
- `stack_layout()`: The layout matrix is used as is, aligning all plots along a single axis.

Examples

```
ggheatmap(matrix(rnorm(81), nrow = 9)) +
    anno_top() +
    ggalign() +
    geom_point(aes(y = value))

# if data is `NULL`, a three column data frame will be created
# (`.panel`, `.index`, `.x`/`.y`)
```

```
ggheatmap(matrix(rnorm(81), nrow = 9)) +
    anno_top(size = 0.5) +
    align_dendro(k = 3L) +
    ggalign(data = NULL, size = 0.2) +
    geom_tile(aes(y = 1L, fill = .panel))
```

`align_group` *Group and align observations based on a group vector*

Description

[Stable]

Splits observations into groups, with slice ordering based on group levels.

Usage

```
align_group(
  group,
  active = NULL,
  set_context = deprecated(),
  name = deprecated()
)
```

Arguments

`group`	A character define the groups of the observations.
`active`	A `active()` object that defines the context settings when added to a layout.
`set_context`	**[Deprecated]** Please use `active` argument instead.
`name`	**[Deprecated]** Please use `active` argument instead.

Value

A `"AlignGroup"` object.

Examples

```
set.seed(1L)
small_mat <- matrix(rnorm(81), nrow = 9)
ggheatmap(small_mat) +
    anno_top() +
    align_group(sample(letters[1:4], ncol(small_mat), replace = TRUE))
```

`align_kmeans` *Split observations by k-means clustering groups.*

Description

[Stable]

Aligns and groups observations based on k-means clustering, enabling observation splits by cluster groups.

Usage

```
align_kmeans(
  centers,
  ...,
  data = NULL,
  active = NULL,
  set_context = deprecated(),
  name = deprecated()
)
```

Arguments

`centers`	either the number of clusters, say k, or a set of initial (distinct) cluster centres. If a number, a random set of (distinct) rows in `x` is chosen as the initial centres.
`...`	Arguments passed on to `stats::kmeans` `iter.max` the maximum number of iterations allowed. `nstart` if `centers` is a number, how many random sets should be chosen? `algorithm` character: may be abbreviated. Note that `"Lloyd"` and `"Forgy"` are alternative names for one algorithm. `trace` logical or integer number, currently only used in the default method (`"Hartigan-Wong"`): if positive (or true), tracing information on the progress of the algorithm is produced. Higher values may produce more tracing information.
`data`	A matrix-like object. By default, it inherits from the layout `matrix`.
`active`	A `active()` object that defines the context settings when added to a layout.
`set_context`	**[Deprecated]** Please use `active` argument instead.
`name`	**[Deprecated]** Please use `active` argument instead.

Value

A `"AlignKmeans"` object.

Axis Alignment for Observations

It is important to note that we consider rows as observations, meaning `vec_size(data)`/`NROW(data)` must match the number of observations along the axis used for alignment (x-axis for a vertical stack layout, y-axis for a horizontal stack layout).

- `quad_layout()`/`ggheatmap()`: For column annotation, the layout `matrix` will be transposed before use (if `data` is a function, it is applied to the transposed matrix), as column annotation uses columns as observations but alignment requires rows.
- `stack_layout()`: The layout matrix is used as is, aligning all plots along a single axis.

Examples

```
ggheatmap(matrix(rnorm(81), nrow = 9)) +
    anno_top() +
    align_kmeans(3L)
```

`align_order` *Order observations based on weights*

Description

[Stable]

Ordering observations based on summary weights or a specified ordering character or integer index.

Usage

```
align_order(
  weights = rowMeans,
  ...,
  reverse = FALSE,
  strict = TRUE,
  data = NULL,
  active = NULL,
  set_context = deprecated(),
  name = deprecated()
)
```

Arguments

`weights`	A summary function which accepts a data and returns the weights for each observations. Alternatively, you can provide an ordering index as either an integer or a character. Since characters have been designated as character indices, if you wish to specify a function name as a string, you must enclose it with `I()`.
`...`	<dyn-dots> Additional arguments passed to function provided in `weights` argument.
`reverse`	A boolean value. Should the sort order be in reverse?

strict	A boolean value indicates whether the order should be strict. If previous groups has been established, and strict is FALSE, this will reorder the observations in each group.
data	A matrix, data frame, or atomic vector used as the input for the weights function. Alternatively, you can specify a function (including purrr-like lambda syntax) that will be applied to the layout matrix, transforming it as necessary for weight calculations. By default, it will inherit from the layout matrix.
active	A active() object that defines the context settings when added to a layout.
set_context	**[Deprecated]** Please use active argument instead.
name	**[Deprecated]** Please use active argument instead.

Value

A "AlignOrder" object.

Axis Alignment for Observations

It is important to note that we consider rows as observations, meaning vec_size(data)/NROW(data) must match the number of observations along the axis used for alignment (x-axis for a vertical stack layout, y-axis for a horizontal stack layout).

- quad_layout()/ggheatmap(): For column annotation, the layout matrix will be transposed before use (if data is a function, it is applied to the transposed matrix), as column annotation uses columns as observations but alignment requires rows.
- stack_layout(): The layout matrix is used as is, aligning all plots along a single axis.

Examples

```
ggheatmap(matrix(rnorm(81), nrow = 9)) +
    anno_left() +
    align_order(I("rowMeans"))
```

---

align_plots *Arrange multiple plots into a grid*

Description

Arrange multiple plots into a grid

Usage

```
align_plots(
  ...,
  ncol = NULL,
  nrow = NULL,
  byrow = TRUE,
  widths = NA,
```

```
  heights = NA,
  design = NULL,
  guides = waiver(),
  theme = NULL
)
```

Arguments

...	<dyn-dots> A list of plots, ususally the ggplot object. Use NULL to indicate an empty spacer.
ncol, nrow	The dimensions of the grid to create - if both are NULL it will use the same logic as facet_wrap() to set the dimensions
byrow	If FALSE the plots will be filled in in column-major order.
widths, heights	The relative widths and heights of each column and row in the grid. Will get repeated to match the dimensions of the grid. The special value of NA will behave as 1null unit unless a fixed aspect plot is inserted in which case it will allow the dimension to expand or contract to match the aspect ratio of the content.
design	Specification of the location of areas in the layout. Can either be specified as a text string or by concatenating calls to area() together.
guides	A string with one or more of "t", "l", "b", and "r" indicating which side of guide legends should be collected. Defaults to waiver(), which inherits from the parent layout. If there is no parent layout, or if NULL is provided, no guides will be collected.
theme	A theme() used to render the guides, title, subtitle, caption, margins, patch.title, panel.border, and background. If NULL (default), will inherit from the parent layout.

Value

An alignpatches object.

See Also

- layout_design()
- layout_title()
- layout_annotation()

Examples

```
# directly copied from patchwork
p1 <- ggplot(mtcars) +
    geom_point(aes(mpg, disp))
p2 <- ggplot(mtcars) +
    geom_boxplot(aes(gear, disp, group = gear))
p3 <- ggplot(mtcars) +
    geom_bar(aes(gear)) +
    facet_wrap(~cyl)
p4 <- ggplot(mtcars) +
```

```
    geom_bar(aes(carb))
p5 <- ggplot(mtcars) +
    geom_violin(aes(cyl, mpg, group = cyl))

# Either add the plots as single arguments
align_plots(p1, p2, p3, p4, p5)

# Or use bang-bang-bang to add a list
align_plots(!!!list(p1, p2, p3), p4, p5)

# Match plots to areas by name
design <- "#BB
           AA#"
align_plots(B = p1, A = p2, design = design)

# Compare to not using named plot arguments
align_plots(p1, p2, design = design)
```

`align_reorder` *Reorders layout observations based on specific statistics.*

Description

Reorders layout observations based on specific statistics.

Usage

```
align_reorder(
  stat,
  ...,
  reverse = FALSE,
  strict = TRUE,
  data = NULL,
  active = NULL,
  set_context = deprecated(),
  name = deprecated()
)
```

Arguments

`stat`	A summary function which accepts a data and returns the statistic, which we'll call `order2()` to extract the ordering information.
`...`	<dyn-dots> Additional arguments passed to function provided in `stat` argument.
`reverse`	A boolean value. Should the sort order be in reverse?
`strict`	A boolean value indicates whether the order should be strict. If previous groups has been established, and strict is `FALSE`, this will reorder the observations in each group.

data	A matrix, data frame, or atomic vector used as the input for the stat function. Alternatively, you can specify a function (including purrr-like lambda syntax) that will be applied to the layout matrix, transforming it as necessary for statistic calculations. By default, it will inherit from the layout matrix.
active	A active() object that defines the context settings when added to a layout.
set_context	**[Deprecated]** Please use active argument instead.
name	**[Deprecated]** Please use active argument instead.

Details

[Experimental]

The align_reorder() function differs from align_order() in that the wts argument in align_order() must return atomic weights for each observation. In contrast, the stat argument in align_reorder() can return more complex structures, such as hclust or dendrogram, among others.

Typically, you can achieve the functionality of align_reorder() using align_order() by manually extracting the ordering information from the statistic.

Value

A "AlignReorder" object.

Axis Alignment for Observations

It is important to note that we consider rows as observations, meaning vec_size(data)/NROW(data) must match the number of observations along the axis used for alignment (x-axis for a vertical stack layout, y-axis for a horizontal stack layout).

- quad_layout()/ggheatmap(): For column annotation, the layout matrix will be transposed before use (if data is a function, it is applied to the transposed matrix), as column annotation uses columns as observations but alignment requires rows.
- stack_layout(): The layout matrix is used as is, aligning all plots along a single axis.

See Also

order2()

Examples

```
ggheatmap(matrix(rnorm(81), nrow = 9)) +
    anno_left() +
    align_reorder(hclust2)
```

area *Define the plotting areas in* align_plots

Description

This is a small helper used to specify a single area in a rectangular grid that should contain a plot. Objects constructed with area() can be concatenated together with c() in order to specify multiple areas.

Usage

```
area(t, l, b = t, r = l)
```

Arguments

t, b	The top and bottom bounds of the area in the grid
l, r	The left and right bounds of the area int the grid

Details

The grid that the areas are specified in reference to enumerate rows from top to bottom, and coloumns from left to right. This means that t and l should always be less or equal to b and r respectively. Instead of specifying area placement with a combination of area() calls, it is possible to instead pass in a single string

```
areas <- c(area(1, 1, 2, 1),
           area(2, 3, 3, 3))
```

is equivalent to

```
areas < -"A##
          A#B
          ##B"
```

Value

A ggalign_area object.

Examples

```
p1 <- ggplot(mtcars) +
    geom_point(aes(mpg, disp))
p2 <- ggplot(mtcars) +
    geom_boxplot(aes(gear, disp, group = gear))
p3 <- ggplot(mtcars) +
    geom_bar(aes(gear)) +
    facet_wrap(~cyl)
```

```
layout <- c(
    area(1, 1),
    area(1, 3, 3),
    area(3, 1, 3, 2)
)

# Show the layout to make sure it looks as it should
plot(layout)

# Apply it to a alignpatches
align_plots(p1, p2, p3, design = layout)
```

`dendrogram_data` *Dengrogram x and y coordinates*

Description

Dengrogram x and y coordinates

Usage

```
dendrogram_data(
  tree,
  priority = "right",
  center = FALSE,
  type = "rectangle",
  leaf_pos = NULL,
  leaf_braches = NULL,
  reorder_branches = TRUE,
  branch_gap = NULL,
  root = NULL
)
```

Arguments

`tree`	A hclust or a dendrogram object.
`priority`	A string of "left" or "right". if we draw from right to left, the left will override the right, so we take the `"left"` as the priority. If we draw from `left` to `right`, the right will override the left, so we take the `"right"` as priority. This is used by `align_dendro()` to provide support of facet operation in ggplot2.
`center`	A boolean value. if `TRUE`, nodes are plotted centered with respect to the leaves in the branch. Otherwise (default), plot them in the middle of all direct child nodes.
`type`	A string indicates the plot type, `"rectangle"` or `"triangle"`.
`leaf_pos`	The x-coordinates of the leaf node. Must be the same length of the number of observations in `tree`.

leaf_braches	Branches of the leaf node. Must be the same length of the number of observations in tree. Usually come from cutree.
reorder_branches	A single boolean value, indicates whether reorder the provided leaf_braches based on the actual index.
branch_gap	A single numeric value indicates the gap between different branches.
root	A length one string or numeric indicates the root branch.

Value

A list of 2 data.frame. One for node coordinates, another for edge coordinates. node and tree segments edge coordinates contains following columns:

- index: the original index in the tree for the current node
- label: node label text
- x and y: x-axis and y-axis coordinates for current node or the start node of the current edge.
- xend and yend: the x-axis and y-axis coordinates of the terminal node for current edge.
- branch: which branch current node or edge is. You can use this column to color different groups.
- panel: which panel current node is, if we split the plot into panel using facet_grid, this column will show which panel current node or edge is from. Note: some nodes may fall outside panel (between two panels), so there are possible NA values in this column.
- .panel: Similar with panel column, but always give the correct branch for usage of the ggplot facet.
- panel1 and panel2: The panel1 and panel2 variables have the same functionality as panel, but they are specifically for the edge data and correspond to both nodes of each edge.
- leaf: A logical value indicates whether current node is a leaf.

Examples

```
dendrogram_data(hclust(dist(USArrests), "ave"))
```

---

fortify_data_frame *Build a data frame*

Description

This function converts various objects to a data frame.

Usage

```
fortify_data_frame(data, ...)
```

Arguments

data	An object to be converted to a data frame.
...	Arguments passed to methods.

Value

A data frame.

See Also

- fortify_data_frame.default()
- fortify_data_frame.character()
- fortify_data_frame.numeric()
- fortify_data_frame.matrix()

---

fortify_data_frame.character

Build a data frame

Description

When data is an atomic vector, it'll be converted to a data frame with following columns:

- .names: the names for the vector (only applicable if names exist).
- value: the actual value of the vector.

Usage

```
## S3 method for class 'character'
fortify_data_frame(data, ...)
```

Arguments

data	An object to be converted to a data frame.
...	Arguments passed to methods.

Value

A data frame.

See Also

Other fortify_data_frame methods: fortify_data_frame.default(), fortify_data_frame.matrix(), fortify_data_frame.numeric()

fortify_data_frame.default

Build a data frame

Description

By default, it calls `fortify()` to build the data frame.

Usage

```
## Default S3 method:
fortify_data_frame(data, ...)
```

Arguments

`data`	An object to be converted to a data frame.
`...`	Arguments passed to methods.

Value

A data frame.

See Also

Other fortify_data_frame methods: `fortify_data_frame.character()`, `fortify_data_frame.matrix()`, `fortify_data_frame.numeric()`

fortify_data_frame.matrix

Build a data frame

Description

When `data` is a matrix, it will automatically be transformed into a long-form data frame, where each row represents a unique combination of matrix indices and their corresponding values.

Usage

```
## S3 method for class 'matrix'
fortify_data_frame(data, ...)
```

Arguments

`data`	An object to be converted to a data frame.
`...`	Arguments passed to methods.

Value

A data frame.

See Also

Other fortify_data_frame methods: `fortify_data_frame.character()`, `fortify_data_frame.default()`, `fortify_data_frame.numeric()`

`fortify_data_frame.numeric` *Build a data frame*

Description

When `data` is an atomic vector, it'll be converted to a data frame with following columns:

- `.names`: the names for the vector (only applicable if names exist).
- `value`: the actual value of the vector.

Usage

```
## S3 method for class 'numeric'
fortify_data_frame(data, ...)
```

Arguments

`data`	An object to be converted to a data frame.
`...`	Arguments passed to methods.

Value

A data frame.

See Also

Other fortify_data_frame methods: `fortify_data_frame.character()`, `fortify_data_frame.default()`, `fortify_data_frame.matrix()`

fortify_matrix *Build a Matrix*

Description

This function converts various objects into a matrix format.

Usage

```
fortify_matrix(data, ...)
```

Arguments

data	An object to be converted to a matrix.
...	Arguments passed to methods.

Value

A matrix.

See Also

- fortify_matrix.default()
- fortify_matrix.MAF()

fortify_matrix.default *Build a Matrix*

Description

By default, it calls as.matrix() to build a matrix.

Usage

```
## Default S3 method:
fortify_matrix(data, ...)
```

Arguments

data	An object to be converted to a matrix.
...	Arguments passed to methods.

Value

A matrix.

See Also

Other fortify_matrix methods: `fortify_matrix.MAF()`

`fortify_matrix.MAF` *Build a Matrix*

Description

By default, it calls `as.matrix()` to build a matrix.

Usage

```
## S3 method for class 'MAF'
fortify_matrix(
  data,
  ...,
  genes = NULL,
  n_top = NULL,
  remove_empty_samples = TRUE,
  collapse_vars = FALSE
)
```

Arguments

`data`	An object to be converted to a matrix.
`...`	Not used currently.
`genes`	An atomic character defines the genes to draw.
`n_top`	A single number indicates how many top genes to be drawn.
`remove_empty_samples`	A single boolean value indicating whether to drop samples without any genomic alterations.
`collapse_vars`	A single boolean value indicating whether to collapse multiple alterations in the same sample and gene into a single value `"Multi_Hit"`.

Value

A matrix.

ggalign attributes

- `gene_anno`: gene summary informations
- `sample_anno`: sample summary informations
- `n_genes`: Total of genes
- `n_samples`: Total of samples
- `breaks`: factor levels of `Variant_Classification`, if `collapse_vars = TRUE`, `"Multi_Hit"` will be added in the end.

See Also

Other fortify_matrix methods: `fortify_matrix.default()`

`fortify_stack` *Build data for the stack layout*

Description

Build data for the stack layout

Usage

```
fortify_stack(data, ...)
```

Arguments

`data`	Any objects to be plot with `ggstack()`.
`...`	Arguments passed to methods.

Value

A matrix or data.frame used by `ggstack()`.

`free_align` *Free from alignment*

Description

align_plots will try to align plot panels, and every elements of the plot, following functions romove these restrictions:

- `free_align`: if we want to compose plots without alignment of some panel axes (panel won't be aligned). we can wrap the plot with `free_align`.
- `free_border`: If we want to compose plots without alignment of the panel borders (but still align the panels themselves), we can wrap the plot with `free_border`.
- `free_lab`: If we want to compose plots without alignment of the axis title, we can wrap the plot with `free_lab`.
- `free_space`: Removing the ggplot element sizes when aligning.
- `free_vp`: Customize the viewport when aligning.
- `free_guide`: If we want to override the behaviour of the overall guides behaviour, we can wrap the plot with `free_guide`.

Usage

```
free_align(plot, axes = "tlbr")

free_border(plot, borders = "tlbr")

free_guide(plot, guides = "tlbr")

free_lab(plot, labs = "tlbr")

free_space(plot, spaces = "tlbr")

free_vp(plot, x = 0.5, y = 0.5, width = NA, height = NA, ...)
```

Arguments

plot	A ggplot or alignpatches object.
axes	Which axes shouldn't be aligned? A string containing one or more of "t", "l", "b", and "r".
borders	Which border shouldn't be aligned? A string containing one or more of "t", "l", "b", and "r".
guides	A string containing one or more of "t", "l", "b", and "r" indicates which side of guide legends should be collected for the plot. If NULL, no guide legends will be collected.
labs	Which axis labs to be free? A string containing one or more of "t", "l", "b", and "r".
spaces	Which border spaces should be removed? A string containing one or more of "t", "l", "b", and "r".
x	A numeric vector or unit object specifying x-location.
y	A numeric vector or unit object specifying y-location.
width	A numeric vector or unit object specifying width.
height	A numeric vector or unit object specifying height.
...	Arguments passed on to grid::viewport

default.units A string indicating the default units to use if x, y, width, or height are only given as numeric vectors.

just A string or numeric vector specifying the justification of the viewport relative to its (x, y) location. If there are two values, the first value specifies horizontal justification and the second value specifies vertical justification. Possible string values are: "left", "right", "centre", "center", "bottom", and "top". For numeric values, 0 means left alignment and 1 means right alignment.

gp An object of class "gpar", typically the output from a call to the function gpar. This is basically a list of graphical parameter settings.

clip One of "on", "inherit", or "off", indicating whether to clip to the extent of this viewport, inherit the clipping region from the parent viewport,

or turn clipping off altogether. For back-compatibility, a logical value of TRUE corresponds to "on" and FALSE corresponds to "inherit".
May also be a grob (or a gTree) that describes a clipping path or the result of a call to as.path.

mask One of "none" (or FALSE) or "inherit" (or TRUE) or a grob (or a gTree) or the result of call to as.mask. This specifies that the viewport should have no mask, or it should inherit the mask of its parent, or it should have its own mask, as described by the grob.

xscale A numeric vector of length two indicating the minimum and maximum on the x-scale. The limits may not be identical.

yscale A numeric vector of length two indicating the minimum and maximum on the y-scale. The limits may not be identical.

angle A numeric value indicating the angle of rotation of the viewport. Positive values indicate the amount of rotation, in degrees, anticlockwise from the positive x-axis.

layout A Grid layout object which splits the viewport into subregions.

layout.pos.row A numeric vector giving the rows occupied by this viewport in its parent's layout.

layout.pos.col A numeric vector giving the columns occupied by this viewport in its parent's layout.

name A character value to uniquely identify the viewport once it has been pushed onto the viewport tree.

Value

- free_align: A modified version of plot with a free_align class.
- free_border: A modified version of plot with a free_border class.
- free_guide: A modified version of plot with a free_guide class.
- free_lab: A modified version of plot with a free_lab class.
- free_space: A modified version of plot with a free_space class.
- free_vp: A modified version of plot with a free_vp class.

Examples

```
# directly copied from `patchwork`
# Sometimes you have a plot that defies good composition alginment, e.g. due
# to long axis labels
p1 <- ggplot(mtcars) +
    geom_bar(aes(y = factor(gear), fill = factor(gear))) +
    scale_y_discrete(
        "",
        labels = c(
            "3 gears are often enough",
            "But, you know, 4 is a nice number",
            "I would def go with 5 gears in a modern car"
```

```
        )
    )

# When combined with other plots it ends up looking bad
p2 <- ggplot(mtcars) +
    geom_point(aes(mpg, disp))

align_plots(p1, p2, ncol = 1L)

# We can fix this be using `free_align`
align_plots(free_align(p1), p2, ncol = 1L)

# If we still want the panels to be aligned to the right, we can choose to
# free only the left side
align_plots(free_align(p1, axes = "l"), p2, ncol = 1L)

# We could use `free_lab` to fix the layout in a different way
align_plots(p1, free_lab(p2), ncol = 1L)

# `free_border` is similar with `free_lab`, they have a distinction in terms
# of placement on either the top or bottom side of the panel. Specifically,
# the top side contains the `title` and `subtitle`, while the bottom side
# contains the `caption`. free_lab() does not attach these elements in the
# panel area.
p3 <- ggplot(mtcars) +
    geom_point(aes(hp, wt, colour = mpg)) +
    ggtitle("Plot 3")
p_axis_top <- ggplot(mtcars) +
    geom_point(aes(mpg, disp)) +
    ggtitle("Plot axis in top") +
    scale_x_continuous(position = "top")
align_plots(p_axis_top, free_lab(p3))
align_plots(p_axis_top, free_border(p3))

# Another issue is that long labels can occupy much spaces
align_plots(NULL, p1, p2, p2)

# This can be fixed with `free_space`
align_plots(NULL, free_space(p1, "l"), p2, p2)
```

`free_gg` *Add ggplot to layout*

Description

[Experimental]

The `free_gg()` function allows you to incorporate a ggplot object into your layout. Unlike `align_gg()`, which aligns every axis value precisely, `free_gg()` focuses more on layout integration without enforcing strict axis alignment. `ggfree()` is an alias for `free_gg`.

Usage

```
free_gg(..., data = waiver(), size = NULL, active = NULL)
```

Arguments

...	Additional arguments passed to ggplot().
data	A dataset used to initialize a ggplot object. By default, it will inherit from the parent layout if applicable. Alternatively, a pre-defined ggplot object can be provided directly.
size	The relative size of the plot, can be specified as a unit.
active	A active() object that defines the context settings when added to a layout.

Value

A free_gg object.

Examples

```
ggheatmap(matrix(rnorm(56), nrow = 7)) +
    anno_top() +
    align_dendro() +
    ggfree(data = mtcars, aes(wt, mpg)) +
    geom_point()
```

---

geom_pie *Pie charts*

Description

Pie charts

Usage

```
geom_pie(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  clockwise = TRUE,
  steps = 100,
  lineend = "butt",
  linejoin = "round",
  linemitre = 10,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used the override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
...	Other arguments passed on to layer()'s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can *not* be passed through .... Unknown arguments that are not part of the 4 categories below are ignored. • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, colour = "red" or linewidth = 3. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the params. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.

- When constructing a layer using a `stat_*()` function, the `...` argument can be used to pass on parameters to the geom part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.
- The `key_glyph` argument of `layer()` may also be passed on through `....` This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

`clockwise` A single boolean value indicates clockwise or not.

`steps` An integer indicating the number of steps to generate the pie chart radian. Increasing this value results in a smoother pie circular.

`lineend` Line end style (round, butt, square).

`linejoin` Line join style (round, mitre, bevel).

`linemitre` Line mitre limit (number greater than 1).

`na.rm` If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display.

`inherit.aes` If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `borders()`.

new aesthetics

- `angle`: the pie circle angle.
- `angle0`: the initial pie circle angle.
- `radius`: the circle radius.

Aesthetics

`geom_pie()` understands the following aesthetics (required aesthetics are in bold):

- `x`
- `y`
- `angle`
- `alpha`
- `angle0`
- `colour`
- `fill`
- `group`

- linetype
- linewidth
- radius

Learn more about setting these aesthetics in `vignette("ggplot2-specs", package = "ggplot2")`.

Examples

```
ggplot(data.frame(x = 1:10, y = 1:10, value = 1:10 / sum(1:10))) +
    geom_pie(aes(x, y, angle = value * 360))
```

---

ggalignGrob *Generate a plot grob.*

Description

Generate a plot grob.

Usage

```
ggalignGrob(x)
```

Arguments

x	An object to be converted into a grob.

Value

A grob() object.

Examples

```
ggalignGrob(ggplot())
```

ggalign_attr *Get data from the* ggalign *attribute*

Description

This function extracts data from the ggalign attribute retained in the data when rendering quad_layout()/ggheatmap() or stack_layout() object. The ggalign attribute holds supplementary information for input data.

Usage

```
ggalign_attr(x, field = NULL)
```

Arguments

x	Input data for the function used to transform the layout data.
field	A string specifying the particular data to retrieve from the ggalign attribute. If NULL, the entire ggalign attribute will be returned. Commonly, this attribute list is attached by fortify_matrix() or fortify_data_frame() functions (refer to the ggalign attributes section in the documentation for details). For examples, see fortify_matrix.MAF().

Value

The specified data from the ggalign attribute or NULL if it is unavailable.

ggalign_stat *Get the statistics from the layout*

Description

Get the statistics from the layout

Usage

```
ggalign_stat(x, ...)

## S3 method for class 'QuadLayout'
ggalign_stat(x, position, ...)

## S3 method for class 'StackLayout'
ggalign_stat(x, what, ...)
```

Arguments

x	A quad_layout()/ggheatmap() or stack_layout() object.
...	Arguments passed to methods.
position	A string of "top", "left", "bottom", or "right".
what	A single number or string of the plot elements in the stack layout.

Value

The statistics

---

ggoncoplot *Create* OncoPrint *Visualizations from Genetic Alteration Data*

Description

[Experimental]

The ggoncoplot() function generates oncoPrint visualizations that display genetic alterations in a matrix format. This function is especially useful for visualizing complex genomic data, such as mutations, copy number variations, and other genomic alterations in cancer research.

Usage

```
ggoncoplot(
  data = NULL,
  mapping = aes(),
  ...,
  map_width = NULL,
  map_height = NULL,
  reorder_row = reorder_column,
  reorder_column = TRUE,
  width = NA,
  height = NA,
  filling = waiver(),
  theme = NULL,
  active = NULL
)

## Default S3 method:
ggoncoplot(
  data = NULL,
  mapping = aes(),
  ...,
  map_width = NULL,
  map_height = NULL,
  reorder_row = reorder_column,
```

```
  reorder_column = TRUE,
  width = NA,
  height = NA,
  filling = waiver(),
  theme = NULL,
  active = NULL
)
```

Arguments

data	A character matrix which encodes the alterations, you can use regex `[;:,|]` to separate multiple alterations.
mapping	Default list of aesthetic mappings to use for plot. If not specified, must be supplied in each layer added to the plot.
...	Additional arguments passed to `fortify_matrix()`.
map_width, map_height	A named numeric value defines the width/height of each alterations.
reorder_row, reorder_column	A boolean value indicating whether to reorder the rows/columns based on the frequency or characteristics of the alterations.
width, height	The relative width/height of the main plot, can be a `unit` object.
filling	Same as `ggheatmap()`, but only `"tile"` can be used.
theme	A `theme()` used to render the `guides`, `title`, `subtitle`, `caption`, `margins`, `patch.title`, `panel.border`, and `background`. If `NULL` (default), will inherit from the parent `layout`.
active	A `active()` object that defines the context settings when added to a layout.

Details

`ggoncoplot()` is a wrapper around the `ggheatmap()` function, designed to simplify the creation of `OncoPrint`-style visualizations. The function automatically processes the input character matrix by splitting the encoded alterations (delimited by regex `[;:,|]`) into individual genomic events and unnesting the columns for visualization.

Additionally, a predefined reordering function, adapted from https://gist.github.com/armish/564a65ab874a770e2c26, is included to enhance the organization of the alterations.

Value

A `HeatmapLayout` object.

Examples

```
# A simple example from `ComplexHeatmap`
mat <- read.table(textConnection(
    "s1,s2,s3
g1,snv;indel,snv,indel
g2,,snv;indel,snv
```

```
g3,snv,,indel;snv"
), row.names = 1, header = TRUE, sep = ",", stringsAsFactors = FALSE)

ggoncoplot(mat, map_width = c(snv = 0.5), map_height = c(indel = 0.9)) +
    # Note that guide legends from `geom_tile` and `geom_bar` are different.
    # Although they appear similar, the internal mechanisms won't collapse
    # the guide legends. Therefore, we remove the guide legends from
    # `geom_tile`.
    guides(fill = "none") +
    anno_top(size = 0.5) +
    ggalign() +
    geom_bar(aes(fill = value), data = function(x) {
        subset(x, !is.na(value))
    }) +
    anno_right(size = 0.5) +
    ggalign() +
    geom_bar(aes(fill = value), orientation = "y", data = function(x) {
        subset(x, !is.na(value))
    }) &
    scale_fill_brewer(palette = "Dark2", na.translate = FALSE)
```

---

ggwrap *Wrap Arbitrary Graphics to ggplot*

Description

The ggwrap() function allows non-ggplot2 elements to be converted into a compliant representation for use with align_plots(). This is useful for adding any graphics that can be converted into a grob with the patch() method.

Usage

```
ggwrap(plot, ..., align = "panel", on_top = TRUE, clip = TRUE, vp = NULL)
```

Arguments

plot	Any graphic that can be converted into a grob using patch().
...	Additional arguments passed to the patch() method.
align	A string specifying the area to place the plot: "full" for the full area, "plot" for the full plotting area (including the axis label), or "panel" for only the actual area where data is drawn.
on_top	A single boolean value indicates whether the graphic plot should be put front-most. Note: the graphic plot will always put above the background.
clip	A single boolean value indicating whether the grob should be clipped if they expand outside their designated area.
vp	A viewport object, you can use this to define the plot area.

Value

A wrapped_plot object that can be directly placed into align_plots().

See Also

- patch.grob() / patch.gList()
- patch.ggplot()
- patch.patch_ggplot()
- patch.patchwork()
- patch.patch()
- patch.trellis()
- patch.formula() / patch.function()
- patch.recordedplot()
- patch.Heatmap()
- patch.HeatmapList()
- patch.HeatmapAnnotation()
- patch.pheatmap()

Examples

```
library(grid)
ggwrap(rectGrob(gp = gpar(fill = "goldenrod")), align = "full") +
    inset(rectGrob(gp = gpar(fill = "steelblue")), align = "panel") +
    inset(textGrob("Here are some text", gp = gpar(color = "black")),
        align = "panel"
    )
p1 <- ggplot(mtcars) +
    geom_point(aes(mpg, disp)) +
    ggtitle("Plot 1")
align_plots(p1, ggwrap(
    ~ plot(mtcars$mpg, mtcars$disp),
    mar = c(0, 2, 0, 0), bg = NA
))
```

---

hclust2 *Generate Tree Structures with Hierarchical Clustering*

Description

Generate Tree Structures with Hierarchical Clustering

Usage

```
hclust2(
  matrix,
  distance = "euclidean",
  method = "complete",
  use_missing = "pairwise.complete.obs"
)
```

Arguments

matrix	A numeric matrix, or data frame.
distance	A string of distance measure to be used. This must be one of "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski". Correlation coefficient can be also used, including "pearson", "spearman" or "kendall". In this way, 1 - cor will be used as the distance. In addition, you can also provide a dist object directly or a function return a dist object. Use NULL, if you don't want to calculate the distance.
method	A string of the agglomeration method to be used. This should be (an unambiguous abbreviation of) one of "ward.D", "ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (= WPGMA), "median" (= WPGMC) or "centroid" (= UPGMC). You can also provide a function which accepts the calculated distance (or the input matrix if distance is NULL) and returns a hclust object. Alternative, you can supply an object which can be coerced to hclust.
use_missing	An optional character string giving a method for computing covariances in the presence of missing values. This must be (an abbreviation of) one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs". Only used when distance is a correlation coefficient string.

Value

A hclust object.

See Also

- cor()
- dist()
- hclust()

Examples

```
hclust2(dist(USArrests), method = "ward.D")
```

heatmap_layout *Arrange Plots in a Heatmap*

Description

[Stable]

heatmap_layout is a specialized version of quad_alignb(), which simplifies the creation of heatmap plots by integrating essential elements for a standard heatmap layout, ensuring that the appropriate data mapping and visualization layers are automatically applied. ggheatmap is an alias for heatmap_layout.

Usage

```
heatmap_layout(
  data = NULL,
  mapping = aes(),
  ...,
  width = NA,
  height = NA,
  filling = waiver(),
  theme = NULL,
  active = NULL,
  set_context = deprecated(),
  order = deprecated(),
  name = deprecated(),
  guides = deprecated()
)
```

Arguments

data	Default dataset to use for the layout. If not specified, it must be supplied in each plot added to the layout. By default, it will try to inherit from parent layout. If not already a matrix, will be converted to one by fortify_matrix().
mapping	Default list of aesthetic mappings to use for plot. If not specified, must be supplied in each layer added to the plot.
...	Additional arguments passed to fortify_matrix().
width, height	The relative width/height of the main plot, can be a unit object.
filling	A single string of "raster" or "tile" to indicate the filling style. By default, waiver() is used, which means that if the input matrix has more than 20,000 cells (nrow * ncol > 20000), geom_raster() will be used for performance efficiency; for smaller matrices, geom_tile() will be used. To customize the filling style, set this to NULL. For backward compatibility, a single boolean value is acceptable: TRUE means waiver(), and FALSE means NULL.

	By default, the classic heatmap color scheme `scale_fill_gradient2(low = "blue", high = "red")` is utilized for continuous values. You can use the options `"ggalign.heatmap_continuous_fill"` or `"ggalign.heatmap_discrete_fil` to modify the default heatmap body filling color scale. See `scale_fill_continuous()` or `scale_fill_discrete()` for details on option settings.
`theme`	A `theme()` used to render the `guides`, `title`, `subtitle`, `caption`, `margins`, `patch.title`, `panel.border`, and `background`. If `NULL` (default), will inherit from the parent `layout`.
`active`	A `active()` object that defines the context settings when added to a layout.
`set_context`	**[Deprecated]** Please use `active` argument instead.
`order`	**[Deprecated]** Please use `active` argument instead.
`name`	**[Deprecated]** Please use `active` argument instead.
`guides`	**[Deprecated]** Please use `plot_align()` function instead.

Value

A `HeatmapLayout` object.

ggplot2 specification

The data input in `ggheatmap` will be converted into the long formated data frame when drawing. The default mapping will use `aes(.data$.x, .data$.y)`, you can use `mapping` argument to control it. The data in the underlying `ggplot` object contains following columns:

- `.xpanel` and `.ypanel`: the column and row panel
- `.x` and `.y`: the `x` and `y` coordinates
- `.row_names` and `.column_names`: A factor of the row and column names of the original matrix (only applicable when names exist).
- `.row_index` and `.column_index`: the row and column index of the original matrix.
- `value`: the actual matrix value.

Examples

```
ggheatmap(1:10)
ggheatmap(letters)
ggheatmap(matrix(rnorm(81), nrow = 9L))
```

inset *Create a ggplot inset*

Description

Create a ggplot inset

Usage

```
inset(plot, ..., align = "panel", on_top = TRUE, clip = TRUE, vp = NULL)
```

Arguments

plot	Any graphic that can be converted into a grob using patch().
...	Additional arguments passed to the patch() method.
align	A string specifying the area to place the plot: "full" for the full area, "plot" for the full plotting area (including the axis label), or "panel" for only the actual area where data is drawn.
on_top	A single boolean value indicates whether the graphic plot should be put front-most. Note: the graphic plot will always put above the background.
clip	A single boolean value indicating whether the grob should be clipped if they expand outside their designated area.
vp	A viewport object, you can use this to define the plot area.

Value

A patch_inset object, which can be added in ggplot.

See Also

- patch.grob() / patch.gList()
- patch.ggplot()
- patch.patch_ggplot()
- patch.patchwork()
- patch.patch()
- patch.trellis()
- patch.formula() / patch.function()
- patch.recordedplot()
- patch.Heatmap()
- patch.HeatmapList()
- patch.HeatmapAnnotation()
- patch.pheatmap()

Examples

```
library(grid)
p1 <- ggplot(mtcars) +
    geom_point(aes(mpg, disp))
p2 <- ggplot(mtcars) +
    geom_boxplot(aes(gear, disp, group = gear))
p1 + inset(p2, vp = viewport(0.6, 0.6,
    just = c(0, 0), width = 0.4, height = 0.4
))
```

---

is_layout *Reports whether* x *is layout object*

Description

Reports whether x is layout object

Usage

```
is_layout(x)

is_quad_layout(x)

is_stack_layout(x)

is_heatmap_layout(x)

is_ggheatmap(x)
```

Arguments

x	An object to test.

Value

A single boolean value.

Examples

```
is_layout(ggheatmap(1:10))

# for quad_layout()
is_quad_layout(quad_alignb(1:10))
is_quad_layout(quad_alignh(1:10))
is_quad_layout(quad_alignv(1:10))
is_quad_layout(quad_free(mtcars))

# for stack_layout()
```

```
is_stack_layout(stack_align(1:10))
is_stack_layout(stack_free(1:10))

# for heatmap_layout()
is_heatmap_layout(ggheatmap(1:10))
is_ggheatmap(ggheatmap(1:10))
```

`layer_order` *Change the layer adding order*

Description

[Experimental]

This function allows you to change the order in which layers are added to a ggplot.

Usage

```
layer_order(layer, order = 0)
```

Arguments

`layer`	A `layer` `geometry` object to be added.
`order`	An integer indicating the position at which the layer should be added. If `<= 0`, the layer will be added at the beginning. If greater than the number of plot layers, it will be added at the end.

Value

A `layer_order` object.

Examples

```
ggplot(faithfuld, aes(waiting, eruptions)) +
    geom_raster(aes(fill = density)) +
    geom_point(color = "red", size = 1)
ggplot(faithfuld, aes(waiting, eruptions)) +
    geom_raster(aes(fill = density)) +
    layer_order(geom_point(color = "red", size = 1))
```

`layout-operator` *Layout operator*

Description

[Experimental]

- `+`: Adds elements to the active plot in the active layout.
- `&`: Applies elements to all plots in the layout.
- `-`: Adds elements to multiple plots in the layout.

Usage

```
## S4 method for signature 'QuadLayout,ANY'
Ops(e1, e2)

## S4 method for signature 'StackLayout,ANY'
Ops(e1, e2)
```

Arguments

`e1`	A `quad_layout()`/`ggheatmap()` or `stack_layout()` object.
`e2`	An object to be added to the plot.

Details

The + operator is straightforward and should be used as needed.

In order to reduce code repetition `ggalign` provides two operators for adding ggplot elements (geoms, themes, facets, etc.) to multiple/all plots in `quad_layout()`/`ggheatmap()` or `stack_layout()` object: `-` and `&`. See `vignette("operator")` for details.

Value

A modified `Layout` object.

Examples

```
set.seed(123)
small_mat <- matrix(rnorm(56), nrow = 7)
ggheatmap(small_mat) +
    anno_top() +
    ggalign() +
    geom_point(aes(y = value))

# `&` operator apply it to all plots
ggheatmap(small_mat) +
    anno_top() +
    align_dendro() &
```

```
    theme(panel.border = element_rect(
        colour = "red", fill = NA, linewidth = unit(2, "mm")
    ))

# If the active layout is the annotation stack, the `-` operator will only
# add the elements to all plots in the active annotation stack:
ggheatmap(small_mat) +
    scale_fill_viridis_c() +
    anno_left(size = 0.2) +
    align_dendro(aes(color = branch), k = 3L) +
    align_dendro(aes(color = branch), k = 3L) -
    # Modify the the color scales of all plots in the left annotation
    scale_color_brewer(palette = "Dark2")

# If the active layout is the `ggstack()`/`stack_layout()` itself, `-`
# applies the elements to all plots in the layout except the nested
# `ggheatmap()`/`quad_layout()`.
stack_alignv(small_mat) +
    align_dendro() +
    ggtitle("I'm from the parent stack") +
    ggheatmap() +
    # remove any active context
    stack_active() +
    align_dendro() +
    ggtitle("I'm from the parent stack") -
    # Modify the the color scales of all plots in the stack layout except the
    # heatmap layout
    scale_color_brewer(palette = "Dark2") -
    # set the background of all plots in the stack layout except the heatmap
    # layout
    theme(plot.background = element_rect(fill = "red"))
```

`layout_annotation` *Modify components of the layout*

Description

- modify the theme of the layout

Usage

```
layout_annotation(theme = waiver(), ...)
```

Arguments

`theme`	A `theme()` used to render the `guides`, `title`, `subtitle`, `caption`, `margins`, `patch.title`, `panel.border`, and `background`. If `NULL` (default), will inherit from the parent `layout`.
`...`	These dots are for future extensions and must be empty.

Details

- guides, patch.title, panel.border, and background will always be added even for the nested alignpatches object.
- title, subtitle, caption, and margins will be added for the top-level alignpatches object only.

Examples

```
p1 <- ggplot(mtcars) +
    geom_point(aes(mpg, disp))
p2 <- ggplot(mtcars) +
    geom_boxplot(aes(gear, disp, group = gear))
p3 <- ggplot(mtcars) +
    geom_bar(aes(gear)) +
    facet_wrap(~cyl)
align_plots(
    p1 + theme(plot.background = element_blank()),
    p2 + theme(plot.background = element_blank()),
    p3 + theme(plot.background = element_blank())
) +
    layout_annotation(
        theme = theme(plot.background = element_rect(fill = "red"))
    )
```

---

layout_design *Define the grid to compose plots in*

Description

To control how different plots are laid out, you need to add a layout design specification. If you are nesting grids, the layout is scoped to the current nesting level.

Usage

```
layout_design(
  ncol = waiver(),
  nrow = waiver(),
  byrow = waiver(),
  widths = waiver(),
  heights = waiver(),
  design = waiver(),
  guides = NA
)
```

Arguments

ncol, nrow	The dimensions of the grid to create - if both are NULL it will use the same logic as facet_wrap() to set the dimensions
byrow	If FALSE the plots will be filled in in column-major order.
widths, heights	The relative widths and heights of each column and row in the grid. Will get repeated to match the dimensions of the grid. The special value of NA will behave as 1null unit unless a fixed aspect plot is inserted in which case it will allow the dimension to expand or contract to match the aspect ratio of the content.
design	Specification of the location of areas in the layout. Can either be specified as a text string or by concatenating calls to area() together.
guides	A string with one or more of "t", "l", "b", and "r" indicating which side of guide legends should be collected. Defaults to waiver(), which inherits from the parent layout. If there is no parent layout, or if NULL is provided, no guides will be collected.

Value

A layout_design object.

Examples

```
p1 <- ggplot(mtcars) +
    geom_point(aes(mpg, disp))
p2 <- ggplot(mtcars) +
    geom_boxplot(aes(gear, disp, group = gear))
p3 <- ggplot(mtcars) +
    geom_bar(aes(gear)) +
    facet_wrap(~cyl)
align_plots(p1, p2, p3) +
    layout_design(nrow = 1L)
align_plots(p1, p2, p3) +
    layout_design(ncol = 1L)
```

---

layout_title *Annotate the whole layout*

Description

Annotate the whole layout

Usage

```
layout_title(title = waiver(), subtitle = waiver(), caption = waiver())
```

Arguments

title	The text for the title.
subtitle	The text for the subtitle for the plot which will be displayed below the title.
caption	The text for the caption which will be displayed in the bottom-right of the plot by default.

Value

A layout_title object.

Examples

```
p1 <- ggplot(mtcars) +
    geom_point(aes(mpg, disp))
p2 <- ggplot(mtcars) +
    geom_boxplot(aes(gear, disp, group = gear))
p3 <- ggplot(mtcars) +
    geom_bar(aes(gear)) +
    facet_wrap(~cyl)
align_plots(p1, p2, p3) +
    layout_title(title = "I'm title")
```

---

order2 *Ordering Permutation*

Description

order2 returns a permutation which rearranges its first argument into ascending order.

Usage

```
order2(x)

## S3 method for class 'hclust'
order2(x)

## S3 method for class 'dendrogram'
order2(x)

## S3 method for class 'ser_permutation_vector'
order2(x)

## S3 method for class 'ser_permutation'
order2(x)
```

Arguments

x	Any objects can be extracting ordering.

Value

An integer vector unless any of the inputs has `2^31` or more elements, when it is a double vector.

Examples

```
order2(hclust2(matrix(rnorm(100L), nrow = 10L)))
```

`patch.alignpatches` *Convert Object into a Grob*

Description

The `patch()` function is used by `ggwrap()` and `inset()` to convert objects into a `grob`.

Usage

```
## S3 method for class 'alignpatches'
patch(x, ...)
```

Arguments

`x`	An object to be converted into a `grob`.
`...`	Not used currently.

Value

A `grob` object.

See Also

`alignpatches`

Other patch methods: `patch.Heatmap()`, `patch.formula()`, `patch.ggplot()`, `patch.grob()`, `patch.patch()`, `patch.patch_ggplot()`, `patch.patchwork()`, `patch.pheatmap()`, `patch.recordedplot()`, `patch.trellis()`

patch.formula *Convert Object into a Grob*

Description

The patch() function is used by ggwrap() and inset() to convert objects into a grob.

Usage

```
## S3 method for class 'formula'
patch(x, ..., device = NULL, name = NULL)

## S3 method for class '`function`'
patch(x, ..., device = NULL, name = NULL)
```

Arguments

x	An object to be converted into a grob.
...	Graphical Parameters passed on to par().
device	A function that opens a graphics device for grid.echo() to work on. By default this is an off-screen, in-memory device based on the pdf device. This default device may not be satisfactory when using custom fonts.
name	A character identifier.

Value

A grob object.

See Also

plot()

Other patch methods: patch.Heatmap(), patch.alignpatches(), patch.ggplot(), patch.grob(), patch.patch(), patch.patch_ggplot(), patch.patchwork(), patch.pheatmap(), patch.recordedplot(), patch.trellis()

patch.ggplot *Convert Object into a Grob*

Description

The patch() function is used by ggwrap() and inset() to convert objects into a grob.

Usage

```
## S3 method for class 'ggplot'
patch(x, ...)
```

Arguments

x	An object to be converted into a grob.
...	Not used currently.

Value

A grob object.

See Also

ggplot

Other patch methods: patch.Heatmap(), patch.alignpatches(), patch.formula(), patch.grob(), patch.patch(), patch.patch_ggplot(), patch.patchwork(), patch.pheatmap(), patch.recordedplot(), patch.trellis()

---

patch.grob *Convert Object into a Grob*

Description

The patch() function is used by ggwrap() and inset() to convert objects into a grob.

Usage

```
## S3 method for class 'grob'
patch(x, ...)

## S3 method for class 'gList'
patch(x, ...)
```

Arguments

x	An object to be converted into a grob.
...	Not used currently.

Value

A grob object.

See Also

Other patch methods: patch.Heatmap(), patch.alignpatches(), patch.formula(), patch.ggplot(), patch.patch(), patch.patch_ggplot(), patch.patchwork(), patch.pheatmap(), patch.recordedplot(), patch.trellis()

patch.Heatmap *Convert Object into a Grob*

Description

The patch() function is used by ggwrap() and inset() to convert objects into a grob.

Usage

```
## S3 method for class 'Heatmap'
patch(x, ..., device = NULL)

## S3 method for class 'HeatmapList'
patch(x, ..., device = NULL)

## S3 method for class 'HeatmapAnnotation'
patch(x, ..., device = NULL)
```

Arguments

x	An object to be converted into a grob.
...	Additional arguments passed to draw().
device	A function that opens a graphics device for temporary rendering. By default this is an off-screen, in-memory device based on the pdf device, but this default device may not be satisfactory when using custom fonts.

Value

A grob object.

See Also

- Heatmap()
- HeatmapAnnotation()

Other patch methods: patch.alignpatches(), patch.formula(), patch.ggplot(), patch.grob(), patch.patch(), patch.patch_ggplot(), patch.patchwork(), patch.pheatmap(), patch.recordedplot(), patch.trellis()

patch.patch *Convert Object into a Grob*

Description

The patch() function is used by ggwrap() and inset() to convert objects into a grob.

Usage

```
## S3 method for class 'patch'
patch(x, ...)
```

Arguments

x	An object to be converted into a grob.
...	Not used currently.

Value

A grob object.

See Also

patch

Other patch methods: patch.Heatmap(), patch.alignpatches(), patch.formula(), patch.ggplot(), patch.grob(), patch.patch_ggplot(), patch.patchwork(), patch.pheatmap(), patch.recordedplot(), patch.trellis()

patch.patchwork *Convert Object into a Grob*

Description

The patch() function is used by ggwrap() and inset() to convert objects into a grob.

Usage

```
## S3 method for class 'patchwork'
patch(x, ...)
```

Arguments

x	An object to be converted into a grob.
...	Not used currently.

Value

A grob object.

See Also

patchwork

Other patch methods: patch.Heatmap(), patch.alignpatches(), patch.formula(), patch.ggplot(), patch.grob(), patch.patch(), patch.patch_ggplot(), patch.pheatmap(), patch.recordedplot(), patch.trellis()

---

patch.patch_ggplot *Convert Object into a Grob*

Description

The patch() function is used by ggwrap() and inset() to convert objects into a grob.

Usage

```
## S3 method for class 'patch_ggplot'
patch(x, ...)
```

Arguments

x	An object to be converted into a grob.
...	Not used currently.

Value

A grob object.

See Also

- patch_titles()
- inset()
- ggwrap()

Other patch methods: patch.Heatmap(), patch.alignpatches(), patch.formula(), patch.ggplot(), patch.grob(), patch.patch(), patch.patchwork(), patch.pheatmap(), patch.recordedplot(), patch.trellis()

patch.pheatmap *Convert Object into a Grob*

Description

The patch() function is used by ggwrap() and inset() to convert objects into a grob.

Usage

```
## S3 method for class 'pheatmap'
patch(x, ...)
```

Arguments

x	An object to be converted into a grob.
...	Not used currently.

Value

A grob object.

See Also

pheatmap()

Other patch methods: patch.Heatmap(), patch.alignpatches(), patch.formula(), patch.ggplot(), patch.grob(), patch.patch(), patch.patch_ggplot(), patch.patchwork(), patch.recordedplot(), patch.trellis()

patch.recordedplot *Convert Object into a Grob*

Description

The patch() function is used by ggwrap() and inset() to convert objects into a grob.

Usage

```
## S3 method for class 'recordedplot'
patch(x, ..., device = NULL)
```

Arguments

x	An object to be converted into a grob.
...	Not used currently.
device	A function that opens a graphics device for grid.echo() to work on. By default this is an off-screen, in-memory device based on the pdf device. This default device may not be satisfactory when using custom fonts.

Value

A grob object.

See Also

recordPlot()

Other patch methods: patch.Heatmap(), patch.alignpatches(), patch.formula(), patch.ggplot(), patch.grob(), patch.patch(), patch.patch_ggplot(), patch.patchwork(), patch.pheatmap(), patch.trellis()

---

patch.trellis *Convert Object into a Grob*

Description

The patch() function is used by ggwrap() and inset() to convert objects into a grob.

Usage

```
## S3 method for class 'trellis'
patch(x, ..., device = NULL)
```

Arguments

x	An object to be converted into a grob.
...	Arguments passed on to grid::grid.grabExpr warn An integer specifying the amount of warnings to emit. 0 means no warnings, 1 means warn when it is certain that the grab will not faithfully represent the original scene. 2 means warn if there's any possibility that the grab will not faithfully represent the original scene. wrap A logical indicating how the output should be captured. If TRUE, each non-grob element on the display list is captured by wrapping it in a grob. wrap.grobs A logical indicating whether, if we are wrapping elements (wrap=TRUE), we should wrap grobs (or just wrap viewports). width,height Size of the device used for temporary rendering.
device	A function that opens a graphics device for temporary rendering. By default this is an off-screen, in-memory device based on the pdf device, but this default device may not be satisfactory when using custom fonts.

Value

A grob object.

See Also

trellis

Other patch methods: patch.Heatmap(), patch.alignpatches(), patch.formula(), patch.ggplot(), patch.grob(), patch.patch(), patch.patch_ggplot(), patch.patchwork(), patch.pheatmap(), patch.recordedplot()

---

patch_titles *Add patch titles to plot borders*

Description

This function extends ggplot2's title functionality, allowing you to add titles to each border of the plot: top, left, bottom, and right.

Usage

```
patch_titles(
  top = waiver(),
  left = waiver(),
  bottom = waiver(),
  right = waiver()
)
```

Arguments

top, left, bottom, right
: A string specifying the title to be added to the top, left, bottom, and right border of the plot.

Details

You can also use labs() to specify the titles (use arguments "top", "left", "bottom", and "right") for the top, left, bottom, and right borders of the plot.

The appearance and alignment of these patch titles can be customized using theme():

- plot.patch_title/plot.patch_title.*: Controls the text appearance of patch titles. By default, plot.patch_title inherit from plot.title, and settings for each border will inherit from plot.patch_title, with the exception of the angle property, which is not inherited.
- plot.patch_title.position/plot.patch_title.position.*: Determines the alignment of the patch titles. By default, plot.patch_title.position inherit from plot.title.position, and settings for each border will inherit from plot.patch_title. The value "panel" aligns the patch titles with the plot panels. Setting this to "plot" aligns the patch title with the entire plot (excluding margins and plot tags).

Value

A labels object to be added to ggplot.

Examples

```
ggplot(mtcars) +
    geom_point(aes(mpg, disp)) +
    patch_titles(
        top = "I'm top patch title",
        left = "I'm left patch title",
        bottom = "I'm bottom patch title",
        right = "I'm right patch title"
    )
```

`plot_align` *Align Specifications in the Layout*

Description

[Experimental]

The `plot_align()` function defines the align Specifications for plots.

Usage

```
plot_align(guides = NA, free_spaces = NA, free_labs = NA)
```

Arguments

`guides`	A string with one or more of `"t"`, `"l"`, `"b"`, and `"r"` indicating which side of guide legends should be collected. Defaults to `waiver()`, which inherits from the parent layout. If no parent layout, all guides will be collected. If `NULL`, no guides will be collected.
`free_spaces`	A string with one or more of `"t"`, `"l"`, `"b"`, and `"r"` indicating which border spaces should be removed. Defaults to `waiver()`, which inherits from the parent layout. If no parent, the default is `NULL`, meaning no spaces are removed. Usually you want to apply this with the whole layout, instead of individual plots.
`free_labs`	A string with one or more of `"t"`, `"l"`, `"b"`, and `"r"` indicating which axis titles should be free from alignment. Defaults to `waiver()`, which inherits from the parent layout. If no parent layout, no axis titles will be aligned. If `NULL`, all axis titles will be aligned.

Value

A `plot_align` object.

Examples

```
set.seed(123)
mat <- matrix(rnorm(72), nrow = 8)
# used in the layout, define the default action for all plots in the layout
ggheatmap(mat) -
    plot_align(guides = NULL) +
    anno_right() +
    align_dendro(aes(color = branch), k = 3)

# You can also add it for a single plot
ggheatmap(mat) -
    # for all plots in the layout, we default won't collect any guide legends
    plot_align(guides = NULL) +
    # for the heatmap body, we collect guide legends in the right
    # note, the guide legends will be collected to the right side of the
    # layout which will overlap the legends in the right annotation
    plot_align(guides = "r") +
    anno_right() +
    align_dendro(aes(color = branch), k = 3)

# to avoid overlapping, we can also collect the guide legends in the
# right annotation
ggheatmap(mat) -
    plot_align(guides = NULL) +
    plot_align(guides = "r") +
    anno_right() +
    align_dendro(aes(color = branch), k = 3) +
    plot_align(guides = "r")
```

---

plot_data *Plot data Specifications*

Description

[Experimental]

Transforms the plot data. Many functions in this package require a specific data format to align observations, `plot_data()` helps reformat data frames as needed.

Usage

```
plot_data(data, inherit = FALSE)
```

Arguments

`data`	A function to transform the plot data before rendering, referred to as `plot_data`. Acceptable values include: • `NULL`: No action taken. • `waiver()`: Inherits from the parent layout.

- A `function` or purrr-style `formula`: Used to transform the plot data, which should accept a data frame and return a data frame. You can apply this after the parent layout `plot_data` function, using the `inherit` argument.

Use this hook to modify the data for all `geoms` after the layout is created (for matrix data, it has been melted to a long format data frame) but before rendering by `ggplot2`. The returned data must be a data frame for ggplot.

`inherit` A single boolean value indicates whether to apply the parent `plot_data` first and then apply the specified `plot_data` for the plot. Defaults to `FALSE`.

Details

Defaults will attempt to inherit from the parent layout if the actual data is inherited from the parent layout, with one exception: `align_dendro()`, which will not inherit the `plot_data` by default.

`plot_theme` *Plot default theme*

Description

[Experimental]

`plot_theme()` serves as the default theme and will always be overridden by any `theme()` settings applied directly to the plot. The default theme (`plot_theme()`) is applied first, followed by any specific `theme()` settings, even if `theme()` is added before `plot_theme()`.

Usage

```
plot_theme(
  ...,
  line,
  rect,
  text,
  title,
  aspect.ratio,
  axis.title,
  axis.title.x,
  axis.title.x.top,
  axis.title.x.bottom,
  axis.title.y,
  axis.title.y.left,
  axis.title.y.right,
  axis.text,
  axis.text.x,
  axis.text.x.top,
  axis.text.x.bottom,
  axis.text.y,
  axis.text.y.left,
```

```
    axis.text.y.right,
    axis.text.theta,
    axis.text.r,
    axis.ticks,
    axis.ticks.x,
    axis.ticks.x.top,
    axis.ticks.x.bottom,
    axis.ticks.y,
    axis.ticks.y.left,
    axis.ticks.y.right,
    axis.ticks.theta,
    axis.ticks.r,
    axis.minor.ticks.x.top,
    axis.minor.ticks.x.bottom,
    axis.minor.ticks.y.left,
    axis.minor.ticks.y.right,
    axis.minor.ticks.theta,
    axis.minor.ticks.r,
    axis.ticks.length,
    axis.ticks.length.x,
    axis.ticks.length.x.top,
    axis.ticks.length.x.bottom,
    axis.ticks.length.y,
    axis.ticks.length.y.left,
    axis.ticks.length.y.right,
    axis.ticks.length.theta,
    axis.ticks.length.r,
    axis.minor.ticks.length,
    axis.minor.ticks.length.x,
    axis.minor.ticks.length.x.top,
    axis.minor.ticks.length.x.bottom,
    axis.minor.ticks.length.y,
    axis.minor.ticks.length.y.left,
    axis.minor.ticks.length.y.right,
    axis.minor.ticks.length.theta,
    axis.minor.ticks.length.r,
    axis.line,
    axis.line.x,
    axis.line.x.top,
    axis.line.x.bottom,
    axis.line.y,
    axis.line.y.left,
    axis.line.y.right,
    axis.line.theta,
    axis.line.r,
    legend.background,
    legend.margin,
    legend.spacing,
```

```
legend.spacing.x,
legend.spacing.y,
legend.key,
legend.key.size,
legend.key.height,
legend.key.width,
legend.key.spacing,
legend.key.spacing.x,
legend.key.spacing.y,
legend.frame,
legend.ticks,
legend.ticks.length,
legend.axis.line,
legend.text,
legend.text.position,
legend.title,
legend.title.position,
legend.position,
legend.position.inside,
legend.direction,
legend.byrow,
legend.justification,
legend.justification.top,
legend.justification.bottom,
legend.justification.left,
legend.justification.right,
legend.justification.inside,
legend.location,
legend.box,
legend.box.just,
legend.box.margin,
legend.box.background,
legend.box.spacing,
panel.background,
panel.border,
panel.spacing,
panel.spacing.x,
panel.spacing.y,
panel.grid,
panel.grid.major,
panel.grid.minor,
panel.grid.major.x,
panel.grid.major.y,
panel.grid.minor.x,
panel.grid.minor.y,
panel.ontop,
plot.background,
plot.title,
```

```
  plot.title.position,
  plot.subtitle,
  plot.caption,
  plot.caption.position,
  plot.tag,
  plot.tag.position,
  plot.tag.location,
  plot.margin,
  strip.background,
  strip.background.x,
  strip.background.y,
  strip.clip,
  strip.placement,
  strip.text,
  strip.text.x,
  strip.text.x.bottom,
  strip.text.x.top,
  strip.text.y,
  strip.text.y.left,
  strip.text.y.right,
  strip.switch.pad.grid,
  strip.switch.pad.wrap,
  complete = FALSE,
  validate = TRUE
)
```

Arguments

`...`	A `theme()` object or additional element specifications not part of base ggplot2. In general, these should also be defined in the `element tree` argument. `Splicing` a list is also supported.
`line`	all line elements (`element_line()`)
`rect`	all rectangular elements (`element_rect()`)
`text`	all text elements (`element_text()`)
`title`	all title elements: plot, axes, legends (`element_text()`; inherits from `text`)
`aspect.ratio`	aspect ratio of the panel
`axis.title`, `axis.title.x`, `axis.title.y`, `axis.title.x.top`, `axis.title.x.bottom`, `axis.title.y.left`, `axis.title.y.right`	labels of axes (`element_text()`). Specify all axes' labels (`axis.title`), labels by plane (using `axis.title.x` or `axis.title.y`), or individually for each axis (using `axis.title.x.bottom`, `axis.title.x.top`, `axis.title.y.left`, `axis.title.y.right`). `axis.title.*.*` inherits from `axis.title.*` which inherits from `axis.title`, which in turn inherits from `text`
`axis.text`, `axis.text.x`, `axis.text.y`, `axis.text.x.top`, `axis.text.x.bottom`, `axis.text.y.left`, `axis.text.y.right`, `axis.text.theta`, `axis.text.r`	tick labels along axes (`element_text()`). Specify all axis tick labels (`axis.text`), tick labels by plane (using `axis.text.x` or `axis.text.y`), or individually for

each axis (using axis.text.x.bottom, axis.text.x.top, axis.text.y.left, axis.text.y.right). axis.text.*.* inherits from axis.text.* which inherits from axis.text, which in turn inherits from text

axis.ticks, axis.ticks.x, axis.ticks.x.top, axis.ticks.x.bottom, axis.ticks.y, axis.ticks.y.left, axis.ticks.y.right, axis.ticks.theta, axis.ticks.r

tick marks along axes (element_line()). Specify all tick marks (axis.ticks), ticks by plane (using axis.ticks.x or axis.ticks.y), or individually for each axis (using axis.ticks.x.bottom, axis.ticks.x.top, axis.ticks.y.left, axis.ticks.y.right). axis.ticks.*.* inherits from axis.ticks.* which inherits from axis.ticks, which in turn inherits from line

axis.minor.ticks.x.top, axis.minor.ticks.x.bottom, axis.minor.ticks.y.left, axis.minor.ticks.y.right, axis.minor.ticks.theta, axis.minor.ticks.r

minor tick marks along axes (element_line()). axis.minor.ticks.*.* inherit from the corresponding major ticks axis.ticks.*.*.

axis.ticks.length, axis.ticks.length.x, axis.ticks.length.x.top, axis.ticks.length.x.bottom, axis.ticks.length.y, axis.ticks.length.y.left, axis.ticks.length.y.right, axis.ticks.length.theta, axis.ticks.length.r

length of tick marks (unit)

axis.minor.ticks.length, axis.minor.ticks.length.x, axis.minor.ticks.length.x.top, axis.minor.ticks.length.x.bottom, axis.minor.ticks.length.y, axis.minor.ticks.length.y.left, axis.minor.ticks.length.y.right, axis.minor.ticks.length.theta, axis.minor.ticks.length.r

length of minor tick marks (unit), or relative to axis.ticks.length when provided with rel().

axis.line, axis.line.x, axis.line.x.top, axis.line.x.bottom, axis.line.y, axis.line.y.left, axis.line.y.right, axis.line.theta, axis.line.r

lines along axes (element_line()). Specify lines along all axes (axis.line), lines for each plane (using axis.line.x or axis.line.y), or individually for each axis (using axis.line.x.bottom, axis.line.x.top, axis.line.y.left, axis.line.y.right). axis.line.*.* inherits from axis.line.* which inherits from axis.line, which in turn inherits from line

legend.background

background of legend (element_rect(); inherits from rect)

legend.margin the margin around each legend (margin())

legend.spacing, legend.spacing.x, legend.spacing.y

the spacing between legends (unit). legend.spacing.x & legend.spacing.y inherit from legend.spacing or can be specified separately

legend.key background underneath legend keys (element_rect(); inherits from rect)

legend.key.size, legend.key.height, legend.key.width

size of legend keys (unit); key background height & width inherit from legend.key.size or can be specified separately

`legend.key.spacing`, `legend.key.spacing.x`, `legend.key.spacing.y`
: spacing between legend keys given as a `unit`. Spacing in the horizontal (x) and vertical (y) direction inherit from `legend.key.spacing` or can be specified separately.

`legend.frame`
: frame drawn around the bar (`element_rect()`).

`legend.ticks`
: tick marks shown along bars or axes (`element_line()`)

`legend.ticks.length`
: length of tick marks in legend (`unit`)

`legend.axis.line`
: lines along axes in legends (`element_line()`)

`legend.text`
: legend item labels (`element_text()`; inherits from `text`)

`legend.text.position`
: placement of legend text relative to legend keys or bars ("top", "right", "bottom" or "left"). The legend text placement might be incompatible with the legend's direction for some guides.

`legend.title`
: title of legend (`element_text()`; inherits from `title`)

`legend.title.position`
: placement of legend title relative to the main legend ("top", "right", "bottom" or "left").

`legend.position`
: the default position of legends ("none", "left", "right", "bottom", "top", "inside")

`legend.position.inside`
: A numeric vector of length two setting the placement of legends that have the `"inside"` position.

`legend.direction`
: layout of items in legends ("horizontal" or "vertical")

`legend.byrow`
: whether the legend-matrix is filled by columns (`FALSE`, the default) or by rows (`TRUE`).

`legend.justification`
: anchor point for positioning legend inside plot ("center" or two-element numeric vector) or the justification according to the plot area when positioned outside the plot

`legend.justification.top`, `legend.justification.bottom`, `legend.justification.left`, `legend.justification.right`, `legend.justification.inside`
: Same as `legend.justification` but specified per `legend.position` option.

`legend.location`
: Relative placement of legends outside the plot as a string. Can be `"panel"` (default) to align legends to the panels or `"plot"` to align legends to the plot as a whole.

`legend.box`
: arrangement of multiple legends ("horizontal" or "vertical")

`legend.box.just`
: justification of each legend within the overall bounding box, when there are multiple legends ("top", "bottom", "left", or "right")

legend.box.margin
: margins around the full legend area, as specified using margin()

legend.box.background
: background of legend area (element_rect(); inherits from rect)

legend.box.spacing
: The spacing between the plotting area and the legend box (unit)

panel.background
: background of plotting area, drawn underneath plot (element_rect(); inherits from rect)

panel.border
: border around plotting area, drawn on top of plot so that it covers tick marks and grid lines. This should be used with fill = NA (element_rect(); inherits from rect)

panel.spacing, panel.spacing.x, panel.spacing.y
: spacing between facet panels (unit). panel.spacing.x & panel.spacing.y inherit from panel.spacing or can be specified separately.

panel.grid, panel.grid.major, panel.grid.minor, panel.grid.major.x, panel.grid.major.y, panel.grid.minor.x, panel.grid.minor.y
: grid lines (element_line()). Specify major grid lines, or minor grid lines separately (using panel.grid.major or panel.grid.minor) or individually for each axis (using panel.grid.major.x, panel.grid.minor.x, panel.grid.major.y, panel.grid.minor.y). Y axis grid lines are horizontal and x axis grid lines are vertical. panel.grid.*.* inherits from panel.grid.* which inherits from panel.grid, which in turn inherits from line

panel.ontop
: option to place the panel (background, gridlines) over the data layers (logical). Usually used with a transparent or blank panel.background.

plot.background
: background of the entire plot (element_rect(); inherits from rect)

plot.title
: plot title (text appearance) (element_text(); inherits from title) left-aligned by default

plot.title.position, plot.caption.position
: Alignment of the plot title/subtitle and caption. The setting for plot.title.position applies to both the title and the subtitle. A value of "panel" (the default) means that titles and/or caption are aligned to the plot panels. A value of "plot" means that titles and/or caption are aligned to the entire plot (minus any space for margins and plot tag).

plot.subtitle
: plot subtitle (text appearance) (element_text(); inherits from title) left-aligned by default

plot.caption
: caption below the plot (text appearance) (element_text(); inherits from title) right-aligned by default

plot.tag
: upper-left label to identify a plot (text appearance) (element_text(); inherits from title) left-aligned by default

plot.tag.position
: The position of the tag as a string ("topleft", "top", "topright", "left", "right", "bottomleft", "bottom", "bottomright") or a coordinate. If a coordinate, can be a numeric vector of length 2 to set the x,y-coordinate relative to the whole plot. The coordinate option is unavailable for plot.tag.location = "margin".

`plot.tag.location`
: The placement of the tag as a string, one of `"panel"`, `"plot"` or `"margin"`. Respectively, these will place the tag inside the panel space, anywhere in the plot as a whole, or in the margin around the panel space.

`plot.margin`
: margin around entire plot (`unit` with the sizes of the top, right, bottom, and left margins)

`strip.background`, `strip.background.x`, `strip.background.y`
: background of facet labels (`element_rect()`; inherits from `rect`). Horizontal facet background (`strip.background.x`) & vertical facet background (`strip.background.y`) inherit from `strip.background` or can be specified separately

`strip.clip`
: should strip background edges and strip labels be clipped to the extend of the strip background? Options are `"on"` to clip, `"off"` to disable clipping or `"inherit"` (default) to take the clipping setting from the parent viewport.

`strip.placement`
: placement of strip with respect to axes, either "inside" or "outside". Only important when axes and strips are on the same side of the plot.

`strip.text`, `strip.text.x`, `strip.text.y`, `strip.text.x.top`, `strip.text.x.bottom`, `strip.text.y.left`, `strip.text.y.right`
: facet labels (`element_text()`; inherits from `text`). Horizontal facet labels (`strip.text.x`) & vertical facet labels (`strip.text.y`) inherit from `strip.text` or can be specified separately. Facet strips have dedicated position-dependent theme elements (`strip.text.x.top`, `strip.text.x.bottom`, `strip.text.y.left`, `strip.text.y.right`) that inherit from `strip.text.x` and `strip.text.y`, respectively. As a consequence, some theme stylings need to be applied to the position-dependent elements rather than to the parent elements

`strip.switch.pad.grid`
: space between strips and axes when strips are switched (`unit`)

`strip.switch.pad.wrap`
: space between strips and axes when strips are switched (`unit`)

`complete`
: set this to `TRUE` if this is a complete theme, such as the one returned by `theme_grey()`. Complete themes behave differently when added to a ggplot object. Also, when setting `complete = TRUE` all elements will be set to inherit from blank elements.

`validate`
: `TRUE` to run `validate_element()`, `FALSE` to bypass checks.

Theme inheritance

Theme elements inherit properties from other theme elements hierarchically. For example, `axis.title.x.bottom` inherits from `axis.title.x` which inherits from `axis.title`, which in turn inherits from `text`. All text elements inherit directly or indirectly from `text`; all lines inherit from `line`, and all rectangular objects inherit from `rect`. This means that you can modify the appearance of multiple elements by setting a single high-level component.

Learn more about setting these aesthetics in `vignette("ggplot2-specs")`.

See Also

`+.gg()` and %+replace%, `element_blank()`, `element_line()`, `element_rect()`, and `element_text()` for details of the specific theme elements.

The modifying theme components and theme elements sections of the online ggplot2 book.

Examples

```
set.seed(123)
small_mat <- matrix(rnorm(56), nrow = 8)
ggheatmap(small_mat) +
    plot_theme(plot.background = element_rect(fill = "red"))

# `plot_theme()` serves as the default theme and will always be
# overridden by any `theme()` settings applied directly to the plot
ggheatmap(small_mat) +
    theme(plot.background = element_rect(fill = "blue")) +
    plot_theme(plot.background = element_rect(fill = "red"))
```

---

quad_active *Determine the Active Context of Quad-Layout*

Description

[Stable]

- quad_active: Sets the active context to the quad_layout()/ggheatmap() itself.
- quad_anno: Sets the active context to the specified annotation stack based on the position argument.
- anno_top: A special case of quad_anno with position = "top".
- anno_left: A special case of quad_anno with position = "left".
- anno_bottom: A special case of quad_anno with position = "bottom".
- anno_right: A special case of quad_anno with position = "right".

Usage

```
quad_active(width = NULL, height = NULL)

quad_anno(position, size = NULL, free_guides = waiver(), what = waiver())

anno_top(size = NULL, free_guides = waiver(), what = waiver())

anno_left(size = NULL, free_guides = waiver(), what = waiver())

anno_bottom(size = NULL, free_guides = waiver(), what = waiver())

anno_right(size = NULL, free_guides = waiver(), what = waiver())
```

Arguments

width, height	The relative width/height of the main plot, can be a unit object.
position	A string of "top", "left", "bottom", or "right" indicates which annotation stack should be activated.
size	A numeric value or an unit object to set the total height/width of the annotation stack. • If position is "top" or "bottom", size sets the total height of the annotation. • If position is "left" or "right", size sets the total width of the annotation.
free_guides	Override the guides collection behavior specified in the quad_layout()/ggheatmap() for the annotation stack.
what	What should get activated in the annotation stack? A single number or string of the plot elements in the stack layout. If NULL, will remove any active context.

Details

By default, quad_anno() will try to initialize the annotation stack layout using data from quad_layout()/ggheatmap(). However, there are situations where the annotation stack cannot be initialized due to incompatible data formats between quad_layout() and the required format for the annotation stack. This often occurs in quad_alignh() and quad_alignv(), where the layout data is a matrix, but top and bottom annotations (in quad_alignh()) or left and right annotations (in quad_alignv()) require a data frame. In such cases, you must use quad_init() to manually initialize the annotation stack.

Value

An object that can be added to quad_layout()/ggheatmap().

See Also

- quad_switch()
- quad_init()

---

quad_free *Arrange Plots in the Quad-Side of a main plot*

Description

[Experimental]

These functions arrange plots around a main plot, allowing for flexible alignment of observations in different directions. ggside is an alias for quad_free.

- quad_free/ggside: Never align observations.
- quad_alignh: Align observations in the horizontal direction.
- quad_alignv: Align observations in the vertical direction.
- quad_alignb: Align observations in both horizontal and vertical directions.

Usage

```
quad_free(
  data = NULL,
  mapping = aes(),
  ...,
  theme = NULL,
  active = NULL,
  width = NA,
  height = NA
)

quad_alignh(
  data = NULL,
  mapping = aes(),
  ...,
  theme = NULL,
  active = NULL,
  width = NA,
  height = NA
)

quad_alignv(
  data = NULL,
  mapping = aes(),
  ...,
  theme = NULL,
  active = NULL,
  width = NA,
  height = NA
)

quad_alignb(
  data = NULL,
  mapping = aes(),
  ...,
  theme = NULL,
  active = NULL,
  width = NA,
  height = NA
)
```

Arguments

data	Default dataset to use for the layout. If not specified, it must be supplied in each plot added to the layout. By default, it will try to inherit from parent layout. • For quad_free/ggside, the function uses fortify_data_frame() to convert the data into a data frame.

- For all other functions, it employs `fortify_matrix()` to convert the data into a matrix.

`mapping` Default list of aesthetic mappings to use for plot. If not specified, must be supplied in each layer added to the plot.

`...` Additional arguments passed to `fortify_matrix()` or `fortify_data_frame()`.

`theme` A `theme()` used to render the `guides`, `title`, `subtitle`, `caption`, `margins`, `patch.title`, `panel.border`, and `background`. If `NULL` (default), will inherit from the parent `layout`.

`active` A `active()` object that defines the context settings when added to a layout.

`width, height` The relative width/height of the main plot, can be a `unit` object.

ggplot2 specification

For `quad_alignb`, `quad_alignh`, and `quad_alignv`, the data input will be converted into the long formated data frame when drawing. The data in the underlying `ggplot` object contains following columns:

- `.xpanel` and `.ypanel`: the column and row panel
- `.x` and `.y`: the `x` and `y` coordinates
- `.row_names` and `.column_names`: A factor of the row and column names of the original matrix (only applicable when names exist).
- `.row_index` and `.column_index`: the row and column index of the original matrix.
- `value`: the actual matrix value.

---

quad_init *Initialize Quad-Layout Annotation*

Description

[Experimental]

Initializes an annotation stack with a user-specified data.

Usage

```
quad_init(position, data = waiver(), ...)
```

Arguments

`position` A string of `"top"`, `"left"`, `"bottom"`, or `"right"` indicates which annotation stack should be initialized.

`data` Default dataset to use for the annotation stack. If not specified, a dataset must be provided for each plot added to the layout. Possible values:

- `waiver()`: try to inherit from the `quad_layout()`.
- `NULL`: no data for the annotation stack.

- Any data which can be coerced by `fortify_matrix()`/`fortify_data_frame()`

Data conversion depends on whether the annotation stack will align the observations:

- If aligned, `fortify_matrix()` will be applied to convert the data into a matrix.
- If not aligned, `fortify_data_frame()` will be used to convert the data into a data frame.

`...` Additional arguments passed to `fortify_matrix()` or `fortify_data_frame()`.

`quad_layout` *Arrange Plots in the Quad-Side of a main plot*

Description

[Experimental]

This function integrates the functionalities of `quad_free()`, `quad_alignh()`, `quad_alignv()`, and `quad_alignb()` into a single interface.

Usage

```
quad_layout(
  data = NULL,
  align = NULL,
  mapping = aes(),
  ...,
  theme = NULL,
  active = NULL,
  width = NA,
  height = NA
)
```

Arguments

`data` Default dataset to use for the layout. If not specified, it must be supplied in each plot added to the layout. By default, it will try to inherit from parent layout. The conversion of data depends on the `align` argument and will use, `fortify_data_frame()` or `fortify_matrix()`:

- If `align` is `NULL`, a data frame is required. When inherited by the annotation stack, no transposition will be applied.
- If `align` is a string, a matrix is required. When inherited by the column annotation stack, it will be transposed.

`align` A string indicating the alignment direction:

- `"horizontal"`: Align plots horizontally.
- `"vertical"`: Align plots vertically.

	• "both": Align plots in both directions. By default, the function does not align observations.
mapping	Default list of aesthetic mappings to use for plot. If not specified, must be supplied in each layer added to the plot.
...	Additional arguments passed to fortify_matrix() or fortify_data_frame().
theme	A theme() used to render the guides, title, subtitle, caption, margins, patch.title, panel.border, and background. If NULL (default), will inherit from the parent layout.
active	A active() object that defines the context settings when added to a layout.
width, height	The relative width/height of the main plot, can be a unit object.

---

quad_switch *Determine the Active Context of Quad-Layout*

Description

[Stable]

quad_switch() integrates quad_active() and quad_anno() into one function for ease of use. This function allows you to quickly change the active context of the quad_layout() and its annotations.

hmanno is an alias for quad_switch, with additional arguments for backward compatibility

Usage

```
quad_switch(
  position = NULL,
  size = NULL,
  width = NULL,
  height = NULL,
  free_guides = waiver(),
  what = waiver()
)

hmanno(
  position = NULL,
  size = NULL,
  width = NULL,
  height = NULL,
  free_guides = waiver(),
  what = waiver(),
  ...,
  guides = deprecated(),
  free_spaces = deprecated(),
  plot_data = deprecated(),
  theme = deprecated(),
  free_labs = deprecated()
)
```

Arguments

position	A string of "top", "left", "bottom", or "right" indicates which annotation stack should be activated. If NULL, it sets the active context to the quad_layout()/ggheatmap() itself.
size	A numeric value or an unit object to set the total height/width of the annotation stack. • If position is "top" or "bottom", size sets the total height of the annotation. • If position is "left" or "right", size sets the total width of the annotation.
width, height	The relative width/height of the main plot, can be a unit object.
free_guides	Override the guides collection behavior specified in the quad_layout()/ggheatmap() for the annotation stack.
what	What should get activated in the annotation stack? A single number or string of the plot elements in the stack layout. If NULL, will remove any active context.
...	These dots are for future extensions and must be empty.
guides	**[Deprecated]** Please use plot_align() function instead.
free_spaces	**[Deprecated]** Please use plot_align() function instead.
plot_data	**[Deprecated]** Please use plot_data() function instead.
theme	**[Deprecated]** Please use plot_theme() function instead.
free_labs	**[Deprecated]** Please use plot_align() function instead.

Value

An object that can be added to quad_layout()/ggheatmap().

See Also

- quad_active()/quad_anno()
- quad_init()

Examples

```
ggheatmap(matrix(rnorm(81), nrow = 9)) +
    anno_top() +
    align_dendro()
```

read_example *Read Example Data*

Description

This function reads example data from the file. If no file is specified, it returns a list of available example files.

Usage

```
read_example(file = NULL)
```

Arguments

file	A string representing the name of the example file to be read. If NULL, the function will return a list of available example file names.

Value

If file is NULL, returns a character vector of available example file names. Otherwise, returns the contents of the specified example file, read as an R object.

Examples

```
read_example()
```

stack_align *Arrange Plots Horizontally or Vertically*

Description

[Experimental]

The stack_align function aligns observations, while stack_free does not.

Several aliases are provided for convenience:

- stack_alignv: A special case of stack_align that sets direction = "vertical".
- stack_alignh: A special case of stack_align that sets direction = "horizontal".
- stack_freev: A special case of stack_free that sets direction = "vertical".
- stack_freeh: A special case of stack_free that sets direction = "horizontal".

Usage

```
stack_align(data = NULL, direction = NULL, ..., theme = NULL, sizes = NA)

stack_alignv(data = NULL, ...)

stack_alignh(data = NULL, ...)

stack_free(data = NULL, direction = NULL, ..., theme = NULL, sizes = NA)

stack_freev(data = NULL, ...)

stack_freeh(data = NULL, ...)
```

Arguments

data	Default dataset to use for the layout. If not specified, it must be supplied in each plot added to the layout. By default, it will try to inherit from parent layout: • For stack_free, fortify_data_frame() will be used to convert data to a data frame. • For stack_align, fortify_matrix() will be used to convert data to a matrix.
direction	A string indicating the direction of the stack layout, either "horizontal" or "vertical".
...	Additional arguments passed to fortify_data_frame() or fortify_matrix().
theme	A theme() used to render the guides, title, subtitle, caption, margins, patch.title, panel.border, and background. If NULL (default), will inherit from the parent layout.
sizes	A numeric or a unit object of length 3 indicating the relative heights (for direction = "horizontal") or widths (for direction = "vertical").

Examples

```
set.seed(123)
stack_align(matrix(rnorm(56), nrow = 7L), "h") +
    align_dendro()
```

---

stack_layout *Put plots horizontally or vertically*

Description

[Experimental]

This function integrates the functionalities of stack_free() and stack_align() into a single interface. ggstack is an alias for stack_layout.

Usage

```
stack_layout(data = NULL, direction = NULL, type = NULL, ...)
```

Arguments

data	Default dataset to use for the layout. If not specified, it must be supplied in each plot added to the layout. By default, it will try to inherit from parent layout: • If type is align, fortify_data_frame() will be used to get a data frame. • If type is free, fortify_matrix() will be used to get a matrix.
direction	A string indicating the direction of the stack layout, either "horizontal" or "vertical".
type	A string indicating the stack layout type: "align" for aligned plots (stack_align()) or "free" for free stacking (stack_free()).
...	Additional arguments passed to fortify_data_frame() or fortify_matrix().

Value

A StackLayout object.

See Also

- stack_align()
- stack_free()

Examples

```
set.seed(123)
small_mat <- matrix(rnorm(56), nrow = 7L)
stack_layout(small_mat, "h", "align") + align_dendro()

# ggstack is an alias for `stack_layout`
ggstack(small_mat, "h", "align") + align_dendro()

# this is the same with:
stack_align(small_mat, "h") + align_dendro()
```

---

stack_switch *Determine the active context of stack layout*

Description

[Stable]

stack_active is an alias for stack_switch(), which sets what = NULL by default, with additional arguments for backward compatibility.

Usage

```
stack_switch(sizes = NULL, what = waiver())

stack_active(
  sizes = NULL,
  what = NULL,
  ...,
  guides = deprecated(),
  free_spaces = deprecated(),
  plot_data = deprecated(),
  theme = deprecated(),
  free_labs = deprecated()
)
```

Arguments

sizes	A numeric or a unit object of length 3 indicating the relative heights (for direction = "horizontal") or widths (for direction = "vertical").
what	What should get activated for the stack layout? A single number or string of the plot elements in the stack layout. If NULL, will remove any active context, this is useful when the active context is a quad_layout() object, where any align_*() will be added to the quad_layout(). By removing the active context, we can add align_*() into the stack_layout().
...	These dots are for future extensions and must be empty.
guides	**[Deprecated]** Please use plot_align() function instead.
free_spaces	**[Deprecated]** Please use plot_align() function instead.
plot_data	**[Deprecated]** Please use plot_data() function instead.
theme	**[Deprecated]** Please use plot_theme() function instead.
free_labs	**[Deprecated]** Please use plot_align() function instead.

Value

A stack_switch object which can be added to stack_layout().

Examples

```
stack_align(matrix(1:9, nrow = 3L), "h") +
    ggheatmap() +
    # ggheamtap will set the active context, directing following addition
    # into the heatmap plot area. To remove the heatmap active context,
    # we can use `stack_active()` which will direct subsequent addition into
    # the stack
    stack_active() +
    # here we add a dendrogram to the stack.
    align_dendro()
```

theme_ggalign *Theme for Layout Plots*

Description

Default theme for quad_layout()/ggheatmap() or stack_layout() object.

Usage

```
theme_ggalign(...)
```

Arguments

...	Arguments passed on to ggplot2::theme_classic base_size base font size, given in pts. base_family base font family base_line_size base size for line elements base_rect_size base size for rect elements

Details

You can change the default theme using the option "ggalign.default_theme". This option should be set to a function that returns a theme() object.

Value

A theme() object.

Examples

```
# Setting a new default theme
old <- options(ggalign.default_theme = function() theme_bw())

# Creating a heatmap with the new theme
ggheatmap(matrix(rnorm(81), nrow = 9)) +
    anno_top() +
    align_dendro(k = 3L)

# Restoring the old default theme
options(old)
```

`theme_no_axes` *Remove axis elements*

Description

Remove axis elements

Usage

```
theme_no_axes(
  axes = "xy",
  text = TRUE,
  ticks = TRUE,
  title = TRUE,
  line = FALSE
)
```

Arguments

`axes`	Which axes elements should be removed? A string containing one or more of `"t"`, `"l"`, `"b"`, `"r"`, `"x"`, and `"y"`.
`text`	If `TRUE`, will remove the axis labels.
`ticks`	If `TRUE`, will remove the axis ticks.
`title`	If `TRUE`, will remove the axis title.
`line`	If `TRUE`, will remove the axis line.

Value

A `theme()` object.

Examples

```
p <- ggplot() +
    geom_point(aes(x = wt, y = qsec), data = mtcars)
p + theme_no_axes()
p + theme_no_axes("b")
p + theme_no_axes("l")
```

`with_quad` *Modify Context for the* `-` *Operator in* `quad_layout()`

Description

[Experimental]

The `with_quad` function allows you to adjust the context in which the subtraction `-` operator is applied within `quad_layout()`. This function wraps objects to specify their target layout contexts when using `-` in `quad_layout()`.

Usage

```
with_quad(x, position = waiver(), main = NULL)
```

Arguments

`x`	The object to be added to the layout using the `-` operator.
`position`	A string specifying one or more positions- `"t"`, `"l"`, `"b"`, and `"r"`- to indicate the annotation stack context for the operator `-`. By default, `waiver()` is used, which sets the behavior as follows: if the active context in `quad_layout()` is `top` or `bottom`, the operator will also apply to the corresponding `bottom` or `top` annotation, respectively. Similarly, if the context is `left` or `right`, the operator applies to the `right` or `left` annotation. When no annotation stack is active, the context defaults to `NULL`.
`main`	A single boolean value indicating whether `x` should apply to the main plot, used only when `position` is not `NULL`. By default, if `position` is `waiver()` and the active context of `quad_layout()` is an annotation stack or the active context of `stack_layout` is itself, `main` will be set to `TRUE`; otherwise, it defaults to `FALSE`.

Value

The original object with an added attribute that sets the specified layout context.

Examples

```
set.seed(123)
small_mat <- matrix(rnorm(56), nrow = 7)

# when the active context in `ggheatmap()`/`quad_layout()` is set to `top` or
# `bottom`, by wrapping object with `with_quad()`, the `-` operator will
# apply changes not only to that annotation but also to the opposite one
# (i.e., bottom if top is active, and vice versa). The same principle
# applies to the left and right annotation.
ggheatmap(small_mat) +
    scale_fill_viridis_c() +
    anno_left(size = 0.2) +
    align_dendro(aes(color = branch), k = 3L) +
```

```
    # Change the active layout to the left annotation
    anno_top(size = 0.2) +
    align_dendro(aes(color = branch), k = 3L) +
    anno_bottom(size = 0.2) +
    align_dendro(aes(color = branch), k = 3L) -
    # Modify the color scale of all plots in the bottom and the opposite
    # annotation
    # in this way, the `main` argument by default would be `TRUE`
    with_quad(scale_color_brewer(palette = "Dark2", name = "Top and bottom"))

# When the `position` argument is manually set, the
# default value of the `main` argument will be `FALSE`.
ggheatmap(small_mat) +
    scale_fill_viridis_c() +
    anno_left(size = 0.2) +
    align_dendro(aes(color = branch), k = 3L) +
    anno_top(size = 0.2) +
    align_dendro(aes(color = branch), k = 3L) +
    anno_bottom(size = 0.2) +
    align_dendro(aes(color = branch), k = 3L) -
    # Modify the background of all plots in the left and top annotation
    with_quad(theme(plot.background = element_rect(fill = "red")), "tl")
```

Index